4-27-00

A

jc811 U.S. PTO 04/25/00

UTILITY PATENT APPLICATION TRANSMITTAL

(for Noncontinuing, Nonprovisional Applications under 37 C.F.R. §1.53(b))

Attorney Docket No. 68702

jc530 U.S. PTO 09/557690 04/25/00

Box PATENT APPLICATION
Commissioner of Patents and Trademarks
ATTENTION: Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Transmitted herewith for filing under 37 C.F.R. §1.53(b) is the nonprovisional, noncontinuing patent application for:

Title: METHOD AND APPARATUS FOR RECEIVING A PLURALITY OF DIFFERENT CODES AT A PLURALITY OF DIFFERENT FREQUENCIES

First Named Inventor or Application Identifier: Nguyen et al.

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

"Express Mail" Mailing Label Number

EL 225 110 679 US

Date of Deposit April 25, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" Service under 37 CFR §1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Edward Price
(Typed or printed name of person mailing)

Edward Price
(Signature of person mailing)

(X) 37 pages of the specification (including claims) are enclosed.

(X) 11 sheet(s) of drawings are enclosed. ( ) Formal (X) Informal

( ) An executed Oath or Declaration and Power of Attorney naming the actual inventors is enclosed.

(X) The names of persons believed to be the actual inventors are set forth in the enclosed unexecuted Oath or Declaration and Power of Attorney (§1.41(a) and §1.53(b)).

( ) An Assignment(s) of the invention to _____, and cover sheet are enclosed.

( ) A check in the amount of $_____ to cover the fee for recording the assignment(s) is enclosed.

( ) A 37 C.F.R. §3.73(b) Statement is enclosed (where an Assignee seeks to take action in a matter before the Patent Office).

( ) An Information Disclosure Statement is enclosed.

( ) A Form PTO-1449 is enclosed.

( ) _____ References (copies) listed on the Form PTO-1449 are enclosed.

(X) A Return Receipt Postcard is enclosed (MPEP §503).

( ) Priority of application number ___/_____ filed on _____ in _____ is claimed under 35 U.S.C. §119.

( ) A certified copy of the priority document is enclosed.

( ) A MicroFiche Computer Program (Appendix) is enclosed.

( ) A Nucleotide and/or Amino Acid Sequence Submission is enclosed.

( ) A Computer Readable Copy is enclosed.

( ) A Paper Copy (Identical to Computer Copy) is enclosed.

( ) A Statement Verifying Identity of above Copies is enclosed.

(X) The filing fee is calculated below:

Fee Calculation For Claims As Filed

(a) Basic Fee			$ 690.00
(b) Independent Claims	5 - 3 = 2	x $ 78.00 =	$ 156.00
(c) Total Claims	13 - 20 = 0	x $ 18.00 =	$ -
(d) Fee for Multiply Dependent Claims		$260.00	$_____
		Total Filing Fee	$ 846.00

( ) A Statement(s) of Status as Small Entity is enclosed, reducing the Filing Fee by half to: $_____

(X) A check in the amount of $ 846.00 to cover the filing fee is enclosed.

( ) Charge $_____ to Deposit Account No. 06-1135.

( ) The payment of the Filing Fee is to be deferred until the Declaration is filed. Do not charge our Deposit Account.

(X) A separate written request under 37 C.F.R. §1.136(a)(3), which is a general authorization to treat any concurrent or future reply requiring a petition for an extension of time under 37 C.F.R. §1.136(a) for its timely submission as incorporating a petition for an extension of time for the appropriate length of time, is enclosed.

(X) The Commissioner is hereby authorized to charge any additional fees which may be required in this application under 37 C.F.R. §§1.16-1.17 during its entire pendency, or credit any overpayment, to Deposit Account No. 06-1135. Should no proper payment be enclosed herewith, as by a check being in the wrong amount, unsigned, post-dated, otherwise improper or informal or even entirely missing, the Commissioner is authorized to charge the unpaid amount to Deposit Account No. 06-1135. This sheet is filed in triplicate.

Attorney Docket No. 68702

( ) Also enclosed:

(X) Address all future communications to Customer Number 22242.

FITCH, EVEN, TABIN & FLANNERY
Suite 1600
120 South LaSalle Street
Chicago, Illinois 60603-3406
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

April 25, 2000
(Date)

Timothy E. Levstik
Registration No. 30,192

METHOD AND APPARATUS FOR RECEIVING A PLURALITY OF DIFFERENT CODES AT A PLURALITY OF DIFFERENT FREQUENCIES

BACKGROUND OF THE INVENTION

This invention relates generally to radio frequency (RF) receivers and more particularly concerns RF receivers capable of receiving a plurality of different codes at a plurality of different frequencies.

Transmitters and receivers are becoming more and more widely used to control the operation of an ever increasing amount of devices and systems. Originally used for military applications and large-scale broadcasting needs, transmitters and receivers have evolved to such an extent that they are now being used in applications as personal as medical implants. In fact it is becoming almost impossible to go a day without using a device that operates via a transmitter and receiver. For example, our cars, garages, shutters, etc. are all offering control via transmitters and receivers. Every day new systems are being designed to take advantage of the mobility transmitters and receivers offer and the ease with which they make even the most tedious of tasks.

As such, there is currently available a wide variety of transmitters and receivers made by numerous manufacturers. Typically each manufacturer's transmitters and/or receivers operate via codes and frequencies unique to that individual manufacturer. Unfortunately, like all electronics, transmitters and receivers can break and/or become damaged. When this happens, it often becomes necessary to purchase replacement parts. However, since most manufacturers build their products to operate via codes and frequencies unique to themselves, consumers are stuck having to buy replacement parts from the original manufacturer. This limits competition and can often make the cost of the replacement part much higher then it would be if other suppliers were available. In some industries,

universal transmitters are offered for sale which can be used on a variety of products made by a variety of manufacturers. For example, in the garage door operator industry, there are several universal transmitters that are capable of operating a variety of door receivers.

Although several universal transmitters are available and help increase competition and/or the number of available suppliers of replacement parts, there are no such alternatives for receivers. If a receiver breaks or is damaged to the extent it needs to be replaced, the user would have to go and purchase an entirely new system or buy a replacement unit from the same manufacturer of his or her old receiver. This may not always be convenient and could be cost prohibitive.

Receivers that are capable of receiving a plurality of different codes at a plurality of different frequencies would not be limited to use as replacement parts. Indeed many service personnel who both install and repair movable barrier operators would prefer carrying such a receiver because it would reduce the need for having several different brands of receivers in their inventory. In addition, it would reduce the number of receivers the service personnel would need to learn how to operate. The mere fact they would be able to buy a larger quantity of receivers from one manufacturer may also allow them a reduced price per unit or price break of some type.

Accordingly, there is a need for a receiver capable of receiving a plurality of different codes at a plurality of different frequencies. There is also a need for a sensitive receiver that can offer these capabilities at relatively low current. There is a further need for a receiver that can offer these capabilities without amplifying unwanted signals resulting from switching in a high gain RF amplifier circuit.

SUMMARY OF THE INVENTION

A RF receiver embodying the present invention is capable of receiving a plurality of different codes at a plurality of different frequencies. More particularly the RF receiver offers digital frequency control by using a controller driven signal diode to add and/or remove capacitance to and/or from a band pass filter circuit. The RF receiver comprises a front end matching antenna for receiving analog RF input and a low gain amplifier coupled to a tunable bandpass filter. The tunable bandpass filter and low gain amplifier are in turn coupled to a super-regenerative amplifier which increases the sensitivity of the receiver and reduces the amount of current drawn by the component. The super-regenerative receiver is coupled to an active filter which supplies a digital signal to the microprocessor or controller of the receiver circuit.

The receiver has various inputs capable of selecting what type of code (or manufacturer's signal) is to be received by the receiver and selecting the bit pattern that is to be received. In addition, relay mode inputs are provided that allow selection between momentary and continuous operation. If momentary operation is selected the receiver will operate upon receipt of a single receiver actuation signal. However if continuous operation is selected the receiver will output only for as long as receiver actuation signals are continuously received.

Once the type of code selection has been made, the controller determines what frequency the tunable bandpass filter should be set to receive. The controller accomplishes this by forward or reverse biasing a signal diode that is connected to additional capacitance and/or circuitry. When the controller forward biases the signal diode, additional capacitance is placed in parallel with the existing capacitance of the bandpass filter and the filter

is set to receive a smaller frequency. When the controller reverse biases the signal diode, the additional capacitance is almost completely removed from the bandpass filter and a larger frequency is set to be received. More particularly, the signal diode of the bandpass circuitry acts as a solid state switch switching among a variety of frequency circuits. Additional signal diodes and the use of tunable components such as tunable inductors and tunable capacitors allow for additional frequencies to be received by the receiver.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a perspective view of an apparatus for moving a barrier or garage door embodying the present invention;

Fig. 2 is a block diagram of a receiver embodying the present invention;

Fig. 3 is a schematic of the receiver shown in Fig. 2;

Figs. 4A-B are schematic diagrams of the two possible tuning circuits shown in Fig. 2; and

Figs. 5A-G are flowcharts of the software operating within the controller depicted in Fig. 3.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring now to the drawings and especially to Fig. 1, a movable barrier operator embodying the present invention is generally shown therein and identified by reference numeral 10. The movable barrier operator 10 includes a head unit 12 mounted within a garage 14 and is employed for controlling the opening and closing of garage 14. More specifically, the head unit 12 is mounted to the ceiling 16 of the garage 14 and includes a rail 18 extending therefrom with a releasable trolley 20 attached having an

arm 22 extending to a multiple paneled garage door 24 positioned for movement along a pair of door rails 26 and 28. The movable barrier operator 10 transfers the garage door 24 between the closed position illustrated in Fig. 1 and an open or raised position, allowing access to and from the garage 14.

The system includes a hand-held transmitter unit 30 adapted to send signals to an antenna 32 positioned on or extending from the head unit 12 and coupled to a receiver 50 (see Fig. 2) located within the head unit 12. The receiver 50 is capable of receiving a plurality of different frequencies as a plurality of different frequencies, as will be discussed in more detail hereinafter. An external control pad 34 is positioned on the outside of the garage 14 having a plurality of buttons 35 thereon and communicates via radio frequency transmission with the antenna 32 and receiver 50 of the head unit 12. A switch module 39 is mounted on a wall of the garage 14. The switch module 39 is connected to the head unit 12 by a pair of wires 39a. The switch module 39 includes a learn switch 39b, a light switch 39c., a lock switch 39d and a command switch 39e.

An optical emitter 42 and an optical detector 46 are coupled to the head unit 12 by a pair of wires 44 and 48, respectively. The emitter 42 and detector 46 are used to satisfy the requirements of Underwriter's Laboratories, the Consumer Product Safety Commission and the like which require that garage door operators sold in the United States must, when in a closing mode and contacting an obstruction having a height of more than one inch, reverse and open the door in order to prevent damage to property and injury to persons.

Referring now to Fig. 2, in which a block diagram of the receiver 50 is shown, the receiver 50 includes an antenna 52 coupled to a front end matching and low gain amplifier circuitry 54. The antenna 52 receives a RF signal from a transmitter. The front end matching and amplifier

circuitry 54 are in turn coupled to a microprocessor or controller controlled solid state selector circuit switch 56 and feedback circuitry 58. The selector circuit 56 determines what frequency receiver 50 will be resonant for, (i.e., 300 MHz, 310 MHz, n MHz), and connects the appropriate circuitry to the band pass filter 60 for receiving this frequency. As will be discussed further below, this can be achieved by using signal diodes to add/remove discrete components to and from a bandpass filter. The bandpass filter 60 is coupled to a super-regenerative high gain amplifier 62 which in turn is coupled to the feedback circuitry 58 and an active filter 64 for filtering a base band signal. The active filter has an operating bandwidth range of 2 kHz centered at approximately 1 kHz (making it within audio range of bandwidth).

The active filter 64 supplies a digital signal to microprocessor or controller 66. Input logic 68 of controller 66 reads the settings of the bit pattern inputs 70, the configuration inputs 72, and the relay mode inputs 74. The configuration inputs 72 determine what type of code will actuate the receiver, (i.e., manufacturer A's code, B's code, C's code, etc.). The relay mode inputs 74 determine whether the receiver will output based on receipt of a momentary input or a continuous input. If momentary input is selected the controller will output to the transmitter output relays upon receiving a single signal, (i.e., the transmitter button must only be pressed once to make receiver operate). If continuous input is selected the receiver will output upon receiving a actuation signal (i.e., the transmitter button must be held down to keep receiver operating). The bit pattern inputs 70 determine what bit pattern the incoming signal must have in order to operate the receiver. The controller 66 stores the configuration and bit pattern in a registry.

The frequency control output 76 of controller 66 determines what frequency matches the configuration selected

and adjusts the selector 56 accordingly. For example, if manufacturer A's code is transmitted at a frequency of 310 MHz and has been selected by the configuration input 72, the frequency control output 76 directs the solid state switch 56 to switch from the 300 MHz tuning circuit (as shown in Fig. 2) to the 310 MHz tuning circuit so that the receiver 50 is set to receive a 310 MHz signal.

The base band signal input 78 of controller 66 receives the digital signal leaving the active filter 64. The controller 66 then compares the digital input received to the configurations and bit patterns selections stored in the controller 66 to determine if the received signal is the signal the receiver 50 has been programmed for, (i.e., determines if a good packet has been received). If the received signal is not the signal the receiver 50 has been programmed for, the receiver ignores the signal and will not respond. However, if the received signal matches the signal the receiver 50 has been programmed for, then the output control logic signal 80 of controller 66 turns on the output relays 82 for an amount of time specified by the relay mode input 74, (i.e., continuous or momentary).

A schematic diagram of a receiver 50 capable of receiving a plurality of different codes at a plurality of different frequencies is shown in Fig. 3. RF antenna 100 is connected to front end matching antenna circuitry 102, and receives incoming analog RF signals. The front end matching antenna circuitry 102 is coupled to a low gain amplifier 104, which is in turn connected to a feedback loop 106 from a super regenerative amplifier. A tunable band pass filter 108 is connected to the feedback loop 106 and comprises a tunable inductor 110, capacitors 112 and 114, signal diode 116, capacitor 118, tunable capacitor 120, and inductor 122. The operation of tunable band pass filter 108 will be discussed further below. Resistors 124, 126 and 128 provide dc biasing for high gain super regenerative amplifier 130. Basically resistors 124, 126 and 128 control the operating

range of super regenerative amplifier 130 and its operation at the dc biasing point. Capacitors 132 and 134, and inductor 136 are coupled to super regenerative amplifier 130 and control the quench rate of the amplifier (i.e., the on/off of the amplifier 130). The quench rate determines the base band bandwidth of the overall circuit.

A first stage active filter circuit 138 with a 2 kHz bandwidth (which is in the audio range) is coupled to the discrete logic controlling the quench rate. A second stage consisting of an envelope detector comparator circuit 140 is coupled to first stage active filter 138 and serves to clean-up any portion of the signal that was missed in the first stage. The last stage comprises a comparator logic circuit 142 which supplies a digital output to pin P32 of controller 144, (i.e., a logic high or low).

The power supply for the circuit is coupled to pins 4, 5, and 6 of terminal block 146 which supplies an AC or DC voltage. Diodes D4, D5, D6 and D7 make up a full wave bridge rectifier 148 which takes an AC waveform and rectifies it to DC. Capacitors 150 and 152 are coupled to the bridge rectifier 148 and filter out AC noise. A jumper 154 is provided to short out resistor 156 if the voltage at point 158 is 12V so that all 12V will go to the voltage regulator 160. If the voltage at point 158 is 24V, then resistor 156 is not shorted out and is used as a current limiter. Diode 162 serves as a blocker to ensure that only DC voltage is coming into the remainder of the power supply circuit. Resistor 164 is a current limiter for LED 166, which serves as a power supply indicator. If power present LED 166 will light up green. Zener diode 168 is used to ensure that the voltage coming into the voltage regulator 160 is not more than 12V. Capacitors 170 and 172 serve as noise filters, and voltage regulator 160 takes 12V in and outputs 5V. The 12V supply is used to control output relays 174 and 176 and the 5V supply is used for the logic circuit as logic high.

The software executing on the controller 144 (as will be discussed further in Figs. 5A-G), sends out a logic high from pin 24 to multi-position switch 178, and reads back the input on pins 19, 20 and 21 of controller 144. This allows the controller 144 to determine the position of multi-position switch 178, thereby determining what code the controller is to look for as the receiver actuating signal, (i.e., manufacturer A's code, B's code, C's code, etc.). More particularly, pins 19, 20, and 21 of controller 144 are normally pulled low by pull down resistors 180. When the logic high is output from pin 24, one of pins 19, 20, or 21 will return a logic high thereby indicating that the setting of multi-position switch 178 corresponding to that pin has been selected.

The controller will determine what frequency the tunable bandpass filter 108 should be set up to receive at based on the setting of multi-position switch 178. In the embodiment of Fig. 3, the bandpass filter circuitry 108 is set up to receive a frequency between the ranges of 280 MHz to 340 MHz, and is tuned to receive either a 300 MHz signal or a 310 MHz signal. Other components can be used to receive a different range of frequencies. Schematic diagrams of the two possible bandpass filter tuning circuits used in Fig. 3 are also shown in Fig. 4A and B. To tune the bandpass filter 108 from one frequency to another, the controller 144 will either output a logic high from pin P25 or not. If pin P25 is set high, diode 116 is turned on (or becomes forward biased) and adds capacitors 118 and 120 in parallel to capacitor 114. The addition of capacitors 118 and 120 increases the overall capacitance tunable inductor 110 sees in the bandpass filter 108 and lowers the frequency to 300 MHz. This is true because capacitance affects frequency according to the equation $2\pi f=1/(1/^{2}\downarrow LC)$. Where f is the desired frequency, L is the inductance, and C is the equivalent capacitance. In Figs. 3 and 4A-B, the L (or the inductance) is tunable inductor 110 and C (or the equivalent

capacitance seen by inductor 110) is capacitor 112 in series with capacitor 118, which is in parallel with capacitors 118 and 120. Capacitor 118 increases the resolution of the tuning of tunable capacitor 120.

The inductor 182 serves to block any high frequency RF from coming in to and damaging the controller 144. This is so because the impedance of an inductor is approximately equal to the frequency times the inductance, so as the frequency gets higher so does the impedance and the inductor serves to block off the high frequency from coming into the microprocessor from the RF circuitry. The inductor 122 performs a similar function. Note that any undesirable RF frequency making it through the bandpass filter 108 is sent to ground rather than allowing such to travel throughout the circuit where it can potentially get amplified and cause the circuit to work improperly.

If pin P25 is set low, diode 116 is not tuned on (or is reverse biased) and the bandpass filter 108 comprises tunable inductor 110 and capacitor 112 in series with capacitor 114. The equivalent capacitance seen by inductor 110 is now smaller, therefore the frequency will increase. For example, if 310 MHz is the desired frequency for the tuning bandpass filter 108 shown in Fig. 3, diode 115 would be switched off so that the equivalent capacitance seen by inductor 110 is smaller. However, if 300 MHz is the desired frequency for the tuning bandpass filter 108, diode 115 would be switched on so that the equivalent capacitance seen by inductor 110 is larger. When diode 115 is turned off (or reverse biased), there is still a small amount of capacitance in parallel with capacitor 114. This is because diode 115 acts as a small capacitor when off (i.e., approximately 2-3 pF) in series with capacitors 118 and 120 in parallel. This fact must be taken into account when tuning the circuit to operate at different frequencies.

Therefore, diode 116 acts as a solid state switch controlled by controller 144 which switches in and/or out

various discrete components in order to add or decrease the capacitance seen by inductor 110. By doing so, the bandpass filter 108 can be tuned to receive a variety of different frequencies. The equivalent capacitance seen by inductor 110 determines the center frequency of the receivers.

Once the controller 144 has determined what code is to be received and at what frequency, it determines whether the relay output should be momentary or continuous. This is accomplished by reading pins 05 and 06 which are connected to the output relay control jumpers 184 and 186. The position of the jumpers 184 and 174 determine whether momentary or continuous operation has been selected. If continuous operation has been selected, the receiver will output only as long as receiver actuation signals are received, (i.e., constant pressure on transmitter button must be applied to continue having the movable barrier move). However, if momentary operation has been selected, the receiver will output upon receipt of one receiver actuation signal, (i.e., one press must be applied to have the movable barrier open or close). Jumper 184 controls relay 174 output and jumper 186 controls relay 176 output.

After the relay output has been selected, the controller polls the pins connected to the configuration DIP switches 188 and 190 to determine what bit pattern an incoming signal must have before the receiver accepts it as an authorized receiver actuation signal. More particularly, the relay sets pin 23 of controller 144 high, sets the remaining output pins P22, P21, and P20 low, puts P24 into a high impedance mode (so it looks like an open circuit for purposes of input coming back to it) and reads the input of pins P00, P01, P02, P03, and P04 to determine the status of switches 1-5 of DIP switch 188. If a logic high is returned to the input pin, the switch associated with that pin is closed. If a logic zero is returned to the input pin, the switch associated with that pin is closed. In order to eliminate the problems associated with mechanical switch

bouncing, vibration and/or noise, ten consecutive reads of the same data must be made before the controller accepts the input.

Once switches 1-5 of DIP switch 188 has been read, pin 22 of controller 144 is set high, the remaining outputs P23, P21, and P20 are set low, P24 is put into high impedance mode, and input pins P00, P01, P02, P03, and P04 are read to determine whether switches 6-10 of DIP switch 188 are open or closed, (i.e., whether logic ones or zeros are returned). This is repeated for DIP 190 such that pin 21 is set high for controller 144 to determine the position of switches 1-5 of DIP 190, and pin 20 is set high for controller 144 to determine the position of switches 6-10 of DIP 190.

Once all of these settings and readings have been made, the receiver 50 is ready to receive a plurality of different codes at a plurality of different frequencies. As discussed above, the switching from one frequency to another can easily be accomplished by turning on or off a signal diode. When on, the signal diode adds capacitance to the bandpass filter adjusting the frequency to some lower frequency. When the diode is off, the signal diode takes out the additional capacitance out of the bandpass filter adjusting the frequency back to the original or some higher frequency. Additional digital frequency control of a receiver can be achieved by adding signal diodes connected to additional frequency circuitry.

Figs. 5A-G are flow charts of the software executing in controller 144. The main routine 200 of the software initializes the controller settings and will only be performed at the initial startup of the controller 144. In step 202, the I/O parameters of are set, telling the controller which pins are input and which pins are output. Specifically, pins P00-P04 are set as input pins for the DIP switch inputs. Pins P05-P06 are set as input pins for the relay mode inputs, and pins P20-P27 are set as inputs for

the DIP switch logic controls. Pin P32 is set as an input for the RADIO_INTERUPT, and is special because it responds to rising edge and falling edge (not just logic highs or lows). Pins P34 and P35 are set as outputs to the relay output controls. The remaining unused I/O pins or ports are set as high impedance inputs, (meaning they look like open circuit inputs). If the controller is not told what each pin is, it will default to an input and increase the risk that the controller will be damaged. In keeping with the examples used thus far, the software flowchart will be discussed as if only three possible manufacturer codes can be selected (A's, B's, or C's code) and A's code is an eight bit code transmitted at 310 MHz, B's code is a ten bit code transmitted at 300 MHz, and C's code is a ten bit code transmitted at 310 MHz.

In step 204, the interrupts priorities are set for the controller 144. There are two interrupts enabled in the software. The software jumps to a RADIO_INTERUPT located at pin P32 whenever an signal edge is received. A down counter T0_INTERUPT is also used to help the controller keep track of timing. The interrupt priority is set so that RADIO_INTERUPT has a higher priority than the T0_INTERUPT. Step 204 also clears any previously stored interrupts. During this step, the controller also disables RADIO_INTERUPT and T0_INTERUPT.

Once this has been completed, the controller 144 sets the timer parameters. In step 206, the prescaler for T0 is set equal to 25, so that each count in T0 is .05 milliseconds (msec.). Down counter T0 is also set equal to 200. The equation the controller 144 uses to determine elapsed time is: T0=200-20*(time in msec.). T0 is the value stored in the down counter register, (which will decrease from 200 as more time passes).

In step 208, several variables are assigned a numeric value. For example, n is set equal to thirty, good packet is set equal to zero, and bit counter is set equal to

one. N is the memory location where the time values saved when a signal edge is received are stored. After the main routine has finished initializing the controller 144, the controller jumps to the start routine 210.

The start routine 210 is where the controller reads the code and bit-pattern settings and adjusts the bandpass filter according to the frequency associated with the manufacturer code selected. In step 212, the control inputs (including the configuration switch and the relay mode output jumpers) are read. Specifically, input pins P01, P02 and P03 (which are coupled to the configuration multi-position switch) are read and saved in the CONTROL_INPUT register, identifying to the controller what manufacturer's code has been selected, (i.e., whether manufacturer A's, B's or C's code has been selected). The output relay mode jumper settings are also read and saved in the CONTROL_INPUT register, identifying to the controller whether the receiver output should be momentary or continuous.

In step 214, the controller sets the RF channel according to the configuration selection made. For example if the configuration switch indicates that manufacturer B's code has been selected, then pin P25 is set as a high output. This turns diode 116 on (forward biased) and causes the bandpass filter 108 to be set up for 300 MHz. If the configuration switch indicates that manufacturer A's or C's code has been selected, pin P25 is set as a high impedance input and diode 116 is turned off (reverse biased) causing the bandpass filter 108 to be set up for 310 MHz.

The DIP switches indicating what receiver actuation signal bit pattern has been selected are read in step 216. The inputs received from reading the DIP switches are saved in the DIP_SWITCH_ID register. If the configuration selection indicates manufacturer A's code has been selected, (which is an eight bit code), the last two

bits (bits nine and ten) are cleared out of the DIP_SWITCH_ID register.

The controller then moves to step 218 and enables both the RADIO_INTERUPT and the T0_INTERUPT. So now, if an edge is received on pin P32, the controller software will jump to the RADIO_INTERUPT subroutine. (See Fig. 5G). If T0 ever times out, the software will jump to the T0_INTERUPT. If an signal edge is received simultaneously with a T0 time out, the RADIO_INTERUPT will take priority over the T0_INTERUPT. In step 218, the T0 timer begins counting down and the RADIO_INTERUPT is set to rising edge, so that upon receipt of the first rising edge, the software will jump to the RADIO_INTERUPT subroutine. Once this step is complete, the software jumps to the Program Loop routine 236.

The RADIO_INTERUPT and T0_INTERUPT routines are show in Fig. 5G. The RADIO_INTERUPT routine 220 will be jumped to if a RF input falling edge or rising edge is detected, (as is indicated in step 222). This interrupt 220 saves the T0 timer values at the point the rising or falling edges are received and resets the T0 timer. These values allow the controller 144 to calculate what the pulse width time is, (i.e., time capturing of the digital signal). More particularly, the RADIO_INTERUPT routine 220 saves the T0 timer value at the time the rise/fall is received to memory array T0_VALUE[n]. This value will later used later on by the controller during the Data Verification subroutine. The RADIO_INTERUPT resets the 10msec._counter, restarts the T0 timer counter, and toggles the radio edge interrupt. If a falling edge was detected the "bit counter" is set equal to "bit counter + 1". The variable "n" is then set equal to "n + 1", so that the next T0 value representing a rising/falling edge of a signal is stored in a different memory location, and the RADIO_INTERUPT routine 220 is exited.

The T0_INTERUPT 228 will be jumped to if the T0 timer times out (or reaches zero). If no radio interrupt is received, the T0 counter should reach zero every ten msec., (as is indicated in step 230). In step 232 the T0_INTERUPT disables the T0 counter if fifty msec. have elapsed without receiving a signal. If fifty msec. have not elapsed without a signal, but the T0 timer has timed out, the "10msec_counter" is set equal to "10msec_counter + 1". The 10msec_counter represents the multiplier for T0. Once the T0_INTERUPT is complete, the software exits the T0_INTERUPT routine 228.

In the Program Loop routine 236, the controller checks the T0 register to see if the elapsed time is greater than four msec. Four msec. represents the minimum amount of blank time. In step 238, if a received signal does not have four msec. of blank time, it is not a code we are looking for and is ignored. The controller will keep reading the T0 timer registers until a gap of four msec. has elapsed without any radio interrupts.

In step 240, the controller determines whether the code selected was manufacturer A's code (which is an eight bit code) or manufacturer B's or C's code. If A's code has been selected control shifts to step 242 and then to step 244 in which the number of bits received is determined. Since manufacturer A's code is an eight bit code, the controller waits until all eight bits have been received before moving on to packet verification step 246. The signal received from falling edge to falling edge is one bit. If eight bits have not been received, the controller continues to wait until all the bits have been received. If eight bits have been received, the entire packet has been received and the controller moves to step 246 packet verification.

If manufacturer A's code was not selected, manufacturer B's or C's code must have been selected (which are 10 bit codes), and the controller moves to step 248. In

step 250, the controller asks whether ten bits have been received. If ten bits have not been received, the controller continues to wait for all the bits to be received. If ten bits have been received, the entire packet has been received and the controller moves to step 246 and begins verifying the packet.

Once the packet verification step 246 has been reached, the controller moves to step 248 and calls the Data Verification subroutine 252 which verifies the packet bit-by-bit. (See Figs. 5E and F). In step 254 of the Data Verification subroutine 252, the variable "n" is set equal to "n + 1" indicating that we are moving to a new memory location. Then, in step 256, the controller determines whether manufacturer A's code has been selected. If A's code has been selected the controller moves to step 258 and 260. In step 260, the controller determines whether T0_VALUE [n] is greater than 0.3 msec. and smaller than 0.6 msec., (which is the first on-time pulse width or on time for manufacturer A's code). If T0_VALUE [n] is within these parameters, control moves to step 262 and "n" is set equal to "n+1" to move to the next memory location. After the value of "n" has been set, the controller moves to step 264 and determines whether T0_VALUE [n] is greater than 1.2 msec. and smaller than 1.7 msec. (which is the off time for manufacturer A's code). If T0_VALUE [n] is within these parameters, the controller moves to step 266, and rotates the zero bit into RECEIVED_PACKET shift register, (specifying that the logic zero bit has been verified).

If the T0_VALUE [n] does not meet the parameters set forth in steps 260 and /or 264, the controller moves to step 268 to determine if a logic one was received rather than a logic zero. In step 268, the controller determines whether T0_VALUE [n] is greater than 1.2 msec. and less than 1.7 msec. If T0_VALUE [n] falls within this parameter the controller moves to step 270 and sets "n" equal to "n+1", (or moves to the next memory location). Then the controller

moves to step 272 and determines whether T0_VALUE [n] is greater than 0.3 msec. and less than 0.6 msec. If it is, the controller moves to step 274 and shifts or left rotates the one bit into the RECEIVED_PACKET shift register, (indicating that a logic one has been received).

Once the shift register has been updated by steps 266 and/or 274, the controller moves to step 276, setting "bit counter" equal to "bit counter - 1" and determining whether "bit counter-1" is equal to zero. If "bit counter-1" does not equal zero, not all of the bits of the packet have been verified, so the controller goes back to step 258 and repeats this procedure with the next bit of the packet. If "bit counter-1" is equal to zero, the controller moves to step 278 and sets "n" back to thirty. Then the controller exits the data verification subroutine in step 280.

If the T0_VALUE [n] does not meet either of the parameters set forth in steps 268 and 272, the controller moves to step 280 and determines that the received packet does not match the receiver actuation signal selected by the three position switch input. Then the controller moves from step 282 to step 278, sets the value of "n" equal to thirty, and exits the data verification routine in step 280.

If the code determination in step 256 indicates that the code is not that of manufacturer A's, the controller moves to step 284 because a ten bit word has to verified. In step 286 the controller determines if the T0_VALUE [n] of the ten bit word is greater than 0.3 msec. and less than 0.6 msec. (which is the first on-time pulse width or on-time for manufacturer B and C's codes). If T0_VALUE [n] is within these parameters, control moves to step 288 and "n" is set equal to "n+1" to move to the next memory location. After the value of "n" has been set, the controller moves to step 290 and determines whether T0_VALUE [n] is greater than 3.1 msec. and smaller than 3.6 msec. (which is the off time for manufacturer B and C's code). If T0_VALUE [n] is within these parameters, the controller

moves to step 292, and rotates the zero bit into the RECEIVED_PACKET shift register, (specifying that a logic zero bit has been verified for the ten bit word).

If the T0_VALUE [n] does not meet the parameters set forth in steps 286 or 290, the controller moves to step 294 to determine if a logic one was received rather than a logic zero. In step 294, the controller determines whether T0_VALUE [n] is greater than 1.8 msec. and less than 2.2 msec. If T0_VALUE [n] falls within this parameter the controller moves to step 296 and sets "n" equal to "n+1", (or moves to the next memory location). Then the controller moves to step 298 and determines whether T0_VALUE [n] is greater than 1.8 msec. and less than 2.2 msec. If it is, the controller moves to step 300 and shifts or left rotates the one bit into the RECEIVED_PACKET shift register, (indicating that a logic one has been received).

Once the shift register has been updated by steps 292 or 300, the controller moves to step 302, setting "bit counter" equal to "bit counter - 1" and determining whether "bit counter-1" is equal to zero. If "bit counter-1" does not equal zero, not all of the bits of the packet have been verified, so the controller goes back to step 284 and repeats this procedure with the next bit of the packet until all ten bits have been received. If "bit counter-1" is equal to zero, the controller moves to step 304 and sets "n" back to thirty. Then the controller exits the data verification subroutine in step 306.

If the T0_VALUE [n] does not meet either of the parameters set forth in steps 294 or 298, the controller moves to step 308 and determines that the received packet does not match the receiver actuation signal selected by the three position switch inputs. Then the controller moves from step 308 to step 304, sets the value of "n" equal to thirty, and exits the data verification routine in step 306.

Once the packet has been verified through the verification subroutine 252, the controller moves to step

252 in the program loop 236, and determines whether the bit pattern of the received packet matches the bit pattern selected by the DIP switches 188 and 190. If it does match, the controller moves to step 312 and sets the variable "Good packet" equal to "Good packet + 1". (This operation will not take place if Good packet is already equal to two.) The controller then moves to step 314 and determines if the new "Good packet" value equals two. If the "good packet" does not equal two, the controller moves back to the start routine 210 and test the packet over again to confirm whether it is a good or bad packet. The program needs two good packets in a row or two bad packets in a row before it is determined to be good or bad, (i.e., the controller won't throw out a packet based on the receipt of one error, but rather requires a confirmation of the fact the packet is either good or bad). If the "good packet" equals two, the controller moves to the output routine 316.

If the bit pattern of the received packet does not match the bit pattern selected by the DIP switches 188 and 190 in step 310, the controller moves to step 318 and sets the variable "Good packet" equal to "Good packet - 1". (This operation will not take place if Good packet already equals zero.) Once the Good packet has been adjusted in step 310, the controller moves to step 320 and determines if the Good packet variable equals zero. If the Good packet variable equals zero, the controller has confirmed that the received packet does not match the selected receiver actuation signal input settings and moves to the output routine 316. If the Good packet variable does not equal zero, the controller moves back to the start routine 210 and test the packet all over again to confirm whether it is a good or bad packet.

The output routine 316 (Fig. 5D) begins with step 322 and the controller asking whether the "Good packet" is equal to two. If the Good packet is not equal to two, the received signal does not match the receiver actuation signal

input selections, and the controller moves to step 324 and shuts off the receiver's relay output (if on). The controller sets the "Bit counter" equal to one and returns to the start routine 210 to begin receiving a new signal. If the Good packet equals two in step 322, the controller moves to step 326 and turns on the receiver's relay output. Then the controller moves to step 328 and determines if the momentary output mode was selected earlier. If momentary output was not selected, the controller moves to step 332 and sets the Bit counter equal to 1. If momentary output was selected, the controller moves to step 330 and enables a 500 msec. delay. Then the controller moves to step 324 and turns off the receiver relay output. After step 324, the controller moves to step 332 to set the "Bit counter" equal to one and returns to the start routine 210 to receive a new signal. The reason for setting the bit counter to one is so that the controller will know the next bit received is from a new word. A listing of the software executing on the controller is attached in an Appendix hereto, (A1-A12).

Thus it is apparent that there has been provided, in accordance with the invention, a method and apparatus for receiving a plurality of codes at a plurality of different frequencies that fully satisfies the objects, aims, and advantages set forth above. While the invention has been described in conjunction with specific embodiments and methods thereof, it is evident that many alternatives, modification, and variations will be apparent to those skilled in the art in light of the foregoing description. Accordingly, it is intended to embrace all such alternatives, modifications, and variations as fall within the spirit and broad scope of the appended claims.

APPENDIX

```
;---- DEFINE FILES USED ------------------------------------------------------
;  Proper listing of all files being use during compiling of the source code.
;==== BEGIN DEFINE ============================================================
   DEFINE  TRI_CODE
         INCLUDE "TRICODE_DEF.INC"
   SEGMENT TRI_CODE
;==== END DEFINE ==============================================================


;---- IRQ VECTOR DEFINITION --------------------------------------------------
; Define the subroutine that will be called upon interupt (if enabled).
; IRQ3 (P30) will be the only one that is enabled for this project.
; This is the RADIO input interupt.
;==== BEGIN IRQ VECTOR ========================================================
   ORG     %00                              ;Top of program memmory.
      .WORD   RADIO_INTERUPT               ;IRQ0, P32 input from radio input circuit.
      .WORD   IRQ1_VECTOR                  ;IRQ1, P33
      .WORD   IRQ2_VECTOR                  ;IRQ2 vector, P31
      .WORD   IRQ3_VECTOR                  ;IRQ3, P30
      .WORD   T0_INTERUPT                  ;IRQ4, T0 interupt.
      .WORD   T1_INTERUPT                  ;IRQ5, T1 interupt.
;==== END IRQ VECTOR ==========================================================


;==== BEGIN MAIN CODE =========================================================
MAIN:
```

```
    ORG     $0C                         ;Beginning address of ROM code

    ;Initializing WDT Section==========;Must be done during the first 60 instruction cycles
    WDT                                 ;Enable WDT.
    srp     #0FH                            ;Select Working Reg Group0, Expand Reg GroupF
    ld      WDTMR,#00001111B                ;Int RC osc. ~80 msec,WDT on during HALT or STOP.
    clr     RP                              ;Return Register Pointer to Bank0 (standard)
    ;Initializing WDT Section==========;

    DI                              ;Disable all interupt.

    ;Initializing I/O port section======;
    ld    PCON,#pcon_ini                ;Initial value of PCON See TriCodeRx.inc for details.
    ld    P01M,#p01m_ini                ;
    ld    P2M,#p2m_ini                  ;
    ld    P2M_COPY,#p2m_ini             ;Mirroring P2M
    ld    P3M,#p3m_ini                  ;
    clr   P0                            ;
    clr   P2                            ;
    clr   P3                            ;
    ;Initializing I/O port section======;

    ;Initializing INTERUPT section======;
    ld    IPR,#ipr_ini                  ;Priority IRQ0,2,4,1,5,3.
    ld    IRQ,#irq_ini                  ;Clear out all IRQ initially.
    ld    IMR,#imr_ini                  ;Disable all interupts initially.
    ;Initializing INTERUPT section======;

    ;Intializing TIMER section =========;
    ld    TMR,#tmr_ini                  ;Initial values for timer control registers.
    ld    PRE0,#pre0_100usec              ;
    ld    T0,#t0_1msec                  ;
    clr     PRE1                        ;
    clr     T1                          ;
    RESET_MULTIPLIER                    ;clear T0 multiplier.
    ;Intializing TIMER section =========;

      jp    START                       ;Goto Start of program.
;==== END MAIN CODE ================================================================


;*************************************************************************************
;*             BEGIN INTERUPT SERVICE ROUTINES SECTION                               *
;*************************************************************************************
;==== RADIO INTERUPT SERVICE ROUTINE ===============================================
;This routine is the 'brain' of determining the validity of the signal received.
;Input :
;  1. Receiver mode: 'A', 'B' or 'C'..(CONTROL_INPUTS)
;  2. T0 properties: T0_MULTIPLIER
;Output:
;  1. Data bits value for valid signal.
;  2. Reset control bits to indicate invalid data.
;==== BEGIN RADIO INTERUPT SERVICE ROUTINE ==========================================
RADIO_INTERUPT:
   WDT
   push  FLAGS                      ;Save FLAGS register for normal operation.
   ld    T0_VALUE,T0                ;Save T0 value for calculation later.
   RELOAD_T0                        ;Reload the initial value for T0.
   TEN_MSEC_NOT_PASSED              ;
   cp    T0_MULTIPLIER,#012H        ;Check if at least 18 msec has elapsed.
   jp    ugt,FIRST_EDGE             ;If so this is the first edge of the first pulse.
   tm    DATA_FLAGS,#00000010B      ;Test if first edge already detected.
   jp    z,EXIT_RADIO_INT           ;If not then just exit.
NEXT_EDGE:
   RESET_MULTIPLIER                 ;Reset the 1msec multiplier for T0.
   tm    P3,#00000100B              ;Check the present input state of radio's input.
   jp    nz,LOW_PULSE_WIDTH         ;If 'HI' then current pulse width is low.
```

```
    ;===== VERIFYING ONTIME PULSE WIDTH ==========================================
HI_PULSE_WIDTH:
    tm      CONTROL_INPUTS,#00000100B  ;Test for " B " mode.
    jp      nz,LINEAR_ON_TIME_CHECK    ;If it is then goto check its 'on time' width.
       ;==== MULTI/STANLEY PULSE CHECK ============================================
MULTI_ON_TIME_CHECK:                   ;else check " C "/" A "  ontime width.
    cp      T0_MULTIPLIER,#01H         ;Check the 1msec multiplier.
    jp      ugt,NOT_VALID_DATA_BIT     ;if greater than or equal 2 msec then not good.
    jp      z,ATLEAST_1MSEC            ;If equal to 1 then check for logic '1' pulse.
          ;==== LOGIC '0' CHECK ================================================
    cp      T0_VALUE,#07H              ;Is it less than or equal .3 msec.
    jp      ugt,NOT_VALID_DATA_BIT     ;If so then not good data.
    cp      T0_VALUE,#05H              ;Is it more than or equal .6 msec.
    jp      ult,NOT_VALID_DATA_BIT     ;If so then data is no good.
    LOGIC_0                            ;.3 <= WIDTH <= .6 therefor logic '0'.
    jp      GOOD_ON_TIME_MEASURED      ;Goto good on time measured.
          ;==== LOGIC '0' CHECK ================================================

          ;==== LOGIC '1' CHECK ================================================
ATLEAST_1MSEC:
    cp      T0_VALUE,#08H              ;Is it less than or equal 1.2 msec.
    jp      ugt,NOT_VALID_DATA_BIT     ;If so then data is no good.
    cp      T0_VALUE,#04H              ;Is it greater than or equal 1.7 msec.
    jp      ult,NOT_VALID_DATA_BIT     ;if so data is no good.
    LOGIC_1                            ;1.2 <= WIDTH <= 1.7 therefor logic '0'.
    jp      GOOD_ON_TIME_MEASURED      ;Goto good on time measured.
          ;==== LOGIC '1' CHECK ================================================
       ;==== MULTI/STANLEY PULSE CHECK ============================================
LINEAR_ON_TIME_CHECK:

    ;===== VERIFYING ONTIME PULSE WIDTH ==========================================

    ;===== VERIFYING OFFTIME PULSE WIDTH =========================================
LOW_PULSE_WIDTH:
    cp      BIT_COUNTER,#0AH           ;Is bit counter at 10.
    jp      z,NOT_VALID_DATA_BIT       ;If it reaches 10 then there is something wrong
    tm      DATA_FLAGS,#00000001B      ;Test for a good ontime was measured previously.
    jp      z,NOT_VALID_DATA_BIT       ;If it's not then don't bother to check off time.
    tm      CONTROL_INPUTS,#00000100B  ;Test for " B " mode.
    jp      nz,LINEAR_OFF_TIME_CHECK   ;If it is then goto check its 'off time' width.
MULTI_OFF_TIME_CHECK:



LINEAR_OFF_TIME_CHECK:
    ;===== VERIFYING OFFTIME PULSE WIDTH ==========================================

    ;===== GOOD ON TIME MEASURED SECTION ==========================================
GOOD_ON_TIME_MEASURED:
    or      DATA_FLAGS,#00000001B      ;Set good ontime pulse flag bit.
    jp      EXIT_RADIO_INT
    ;===== GOOD ON TIME MEASURED SECTION ==========================================

VALID_DATA_BIT:
    jp      EXIT_RADIO_INT

NOT_VALID_DATA_BIT:
    and     DATA_FLAGS,#11111100B      ;Clear good 'ontime' and the 'first edge' flag bit.
EXIT_RADIO_INT:
    pop       FLAGS
    iret

    ;==== FIRST EDGE INTERUPT SECTION ============================================
FIRST_EDGE:
    RESET_MULTIPLIER                   ;Reset T0 multiplier to zero.
    RESET_BIT_COUNTER                  ;
```

```
   RESET_BIT_1_POSITION              ;
   RESET_BIT_0_POSITION              ;
   or    DATA_FLAGS,#00000010B       ;Set bit1 for first edge detected.
   jp    EXIT_RADIO_INT
   ;==== FIRST EDGE INTERUPT SECTION ==========================================================
;==== END RADIO INTERUPT SERVICE ROUTINE ==================================================


;==== IRQ1_VECTOR INTERUPT SERVICE ROUTINE ================================================
;==== BEGIN IRQ1_VECTOR INTERUPT SERVICE ROUTINE ==========================================
IRQ1_VECTOR
   iret
;==== END IRQ1_VECTOR INTERUPT SERVICE ROUTINE ============================================

;==== IRQ2_VECTOR INTERUPT SERVICE ROUTINE ================================================
;==== BEGIN IRQ2_VECTOR INTERUPT SERVICE ROUTINE ==========================================
IRQ2_VECTOR
   iret
;==== END IRQ2_VECTOR INTERUPT SERVICE ROUTINE ============================================

;==== IRQ3_VECTOR INTERUPT SERVICE ROUTINE ================================================
;==== BEGIN IRQ3_VECTOR INTERUPT SERVICE ROUTINE ==========================================
IRQ3_VECTOR
   iret
;==== END IRQ3_VECTOR INTERUPT SERVICE ROUTINE ============================================
;==== INTERUPT SERVICE ROUTINES SECTION ===================================================

;==== T0_VECTOR INTERUPT SERVICE ROUTINE ==================================================
;Purpose:
;  This interupt capture T0 timer. It is started by the first state change in Radio Int.
;  at pin32 (IRQ0). It is continous counting. It should time out (get here) every 1 msec
;  The T0_MULTIPLIER is then incremented, therefore this register value can be considered
;  as the msec value.The initial value for T0 is 10 (decimal), therefor each count in T0
;  is an equivalent of .1 msec. Example: if T0 = 8 the .2 msec has elapsed.
;Input:
;  T0 reaches end of count.
;Output:
;  T0_MULTIPLIER: Store the # of T0 timeout(msec) without Radio interupt being detected.
;==== BEGIN T0_VECTOR INTERUPT SERVICE ROUTINE ============================================
T0_INTERUPT
   WDT
   push  FLAGS                 ;Save flags
   inc   T0_MULTIPLIER         ;Increment the xmsec counter.
   cp    T0_MULTIPLIER,#0AH    ;Test if 10 msec has elapsed.
   jp    z,NO_PULSE_FOR_10MSEC;If so set the proper bit.
   cp    T0_MULTIPLIER,#023H   ;If 35 msec (23H) has passed then stop counting.
   jp    z,STOP_COUNTING       ;If no signal for 35 msec stop the timer.
   jp    EXIT_T0_INTERUPT      ;
NO_PULSE_FOR_10MSEC:
   TEN_MSEC_PASSED             ;10 msechas passed since last pulse.
STOP_COUNTING:
   DISABLE_T0_COUNT            ;Stop T0 count.
   LD    T0_MULTIPLIER,#0FFH   ;Set all bits for overcount.(no signal for 35msec)
EXIT_T0_INTERUPT:
   pop   FLAGS                 ;restore flags value.
   iret
;==== END T0_VECTOR INTERUPT SERVICE ROUTINE ==============================================

;==== T1_VECTOR INTERUPT SERVICE ROUTINE ==================================================
;==== BEGIN T1_VECTOR INTERUPT SERVICE ROUTINE ============================================
T1_INTERUPT
   WDT
   iret
;==== END T1_VECTOR INTERUPT SERVICE ROUTINE ==============================================
;******************************************************************************************
;*              END INTERUPT SERVICE ROUTINES SECTION                                     *
;******************************************************************************************
```

```
;;==== READ P0 INPUTS ================================================================
;Purpose:
;  1. Read P0 inputs with debounce.Inputs must be the same for 10 consecutive read cycles
;     to be a valid input. If no valid reading can be obtained the WDT will reset the uC.
;General Algorithym:
;     P2 I/O controls which set of P0 input is being read so it's must be know prior to
;  calling this routine. The entire P0 is read minmum of 10 times, each time the previous
;  value is compared to the current value. If they are not the same (noises) then the
;  counter is reset and the process repeat. 10 consecutive identical reading is required for
;  a valid input for P0. If such condition can not be achieved then the WDT should time out
;  and reset the micro. This would avoid a infinite loop.
;Input:
;  1. P2M I/O controls: must be pre-determine by the calling routine.
;Output:
;  1. P0_INPUT: All the content of P0 is stored in P0_INPUT.
;Subroutine called:
;  1. None.
;==== BEGIN READ P0 INPUTS ============================================================
READ_P0:
   WDT
   ld    DEBOUNCE_COUNTER,#debounce_value;load initial value for debounce counter (10)
   ld    OLD_INPUT,P0               ;Read P0 and store it as previous input.
   WDT                              ;
DEBOUNCE_P0:
   ld    NEW_INPUT,P0               ;Read P0 and store as current input.
   cp    OLD_INPUT,NEW_INPUT        ;Compare previous and current input.
   jp    z,P0_SAME                  ;If the same then goto P0_SAME.
P0_NOT_SAME:
   ld    DEBOUNCE_COUNTER,#debounce_value;reload initial value for debounce counter
   jp    SAVE_NEW_INPUT             ;
P0_SAME:
   dec   DEBOUNCE_COUNTER           ;decrement debounce counter.
   jp    z,EXIT_READ_P0             ;If zero then exit
SAVE_NEW_INPUT:
   ld    OLD_INPUT,NEW_INPUT        ;Save current input.
   jp    DEBOUNCE_P0                ;Re-read P0 again.

EXIT_READ_P0:
   ld    P0_INPUT,NEW_INPUT         ;Store new input in P0_INPUT register.
   ret                              ;EXIT
;==== BEGIN READ P0 INPUTS ============================================================


;==== SET RF CHANNEL ==================================================================
;Purpose:
;  Set P25 hi or low according to the mode slide switch input:
;      C            = CHANNEL 1 (300 MHz) => P25 = OUTPUT/LO
;      B  L A       = CHANNEL 2 (310 MHz) => P25 = INPUT/HI Z.
;General Algorithm:
;     Use the input from CONTROL_INPUTS register to determine the out put of P25.
;Input:
;     CONTROL_INPUTS
;Output:
;     P25's state. No other register is affected.
;==== BEGIN SET RF CHANNEL ============================================================
SET_RF_CHANNEL:
   WDT
   tm    CONTROL_INPUTS,#00000010B  ;Test for 'C' input.
   jp    nz,TUNE_TO_300MHz          ;Goto and tune to 300MHz.
TUNE_TO_310MHz:
   or    P2M_COPY,#00100000B        ;Set P25 to input/Hi Z.
   ld    P2M,P2M_COPY               ;
   ret                              ;
TUNE_TO_300MHz:
```

```
   and   P2M_COPY,#11011111B        ;Set P25 to output
   ld    P2M,P2M_COPY               ;
   NOP                              ;Waste 1-cycle for change state.
   or    P2,#00100000B              ;Set out put to lo/GND.
   ret                              ;
;==== END SET RF CHANNEL ==========================================================

;==== READ CONTROL INPUTS ==========================================================
;Purpose:
;  1. Read S3 switches to determine B , C or A mode of operation.
;  2. Read P2 and P4 to determine constant pressure or momentary operation for both channel.
;General Algorithm:
;     Set P24 pin to output and set it HI (5V). Read P0 to determine the slide switch position
;  (: B , C , A ) and the switch mode (constant pressure or momentary). After
;  complete the read reset the output P24 to LO then change it to an input pin (high
;  impedance). The result is stored in CONTROL_INPUT register for future use. This register
;  will not be changed until the next read cycle or this routine is being called. The value of
;  P2M is not being changed. It should be the same after this routine is exited.
;Input:
;  1. None
;Output:
;  2. CONTROL_INPUTS register bits 0,1,2,5,6 will change accordingly other remain undisturb.
;Subroutine called:
;  1. READ_P0.
;==== BEGIN READ CONTROL INPUTS ====================================================
READ_CONTROL_INPUTS:
   WDT
   and   P2M_COPY,#11101111B  ;Using P2M_COPY as intermediate/P2M is write only.
   ld    P2M,P2M_COPY            ;Change only P24 to output undisturb the rest.
   NOP                           ;Wait one cycle before put data on output pin.
   or    P2,#00010000B           ;Set P24 to Hi (Mode select switch).
   call  READ_P0                 ;Read P0 with debounce.
   and   P0_INPUT,#01100111B     ;Mask out all other except for mode switches inputs.
   ld    CONTROL_INPUTS,P0_INPUT ;Store mode switch input to the proper bit location.
   and   P2,#11101111B           ;Shut off P24 output.
   NOP                           ;Wait 1 cycle before I/O change.
   or    P2M_COPY,#00010000B     ;Set P24 to high impedance input.
   ld    P2M,P2M_COPY            ;using P2M_COPY as intermediate register.
   ret                           ;
;==== END READ CONTROLS INPUTS =====================================================

;==== READ DIP SWITCH ==============================================================
;Purpose:
;  1. Read DIP switch input to determine the unique code.
;General Algorithm:
;  1. Determine the channel using CONTROL_INPUT registers bit 0,1,2. It bit 1 is high then
;     it's channel 1 (300Mhz) otherwise it's channel 2.
;  2. Once the channel is determined either S1 or S2 will be read as follow:
;        * DIP switch 1 thru 5 is read and store in DIP_SW_HIBYTE [4-0]. Example: DIP1's
;          value (D1) is stored in bit4 location of register DIP_SW_HIBYTE.
;        * DIP switch 6 thru 10 is read and store in DIP_SW_HIBYTE [4-0].
;  3. Left Rotate DIP_SW_LOWBYTE 3 times to bring D6 to bit7.
;  4. Left Rotate DIP_SW_LOWBYTE through Carry 'C' then left rotate DIP_SW_HIBYTE. Do this
;     3 times.
;  5. The final format of the register should be:
;        DIP_SW_HIBYTE: D1 D2 D3 D4 D5 D6 D7 D8
;        DIP_SW_LOWBYTE:D9 D10 0 0 0 0 0 0
;
;Input:
;  1. CONTROL_INPUTS: to determine which set of DIP switch to read from.
;Output:
;  1. DIP_SW_LOWBYTE, DIP_SW_HIBYTE: Store the ID code read from DIP switches
;Subroutine called:
;  1. READ_P0.
;==== BEGIN READ DIP SWITCH ========================================================
READ_DIP_SWITCH:
```

```
   WDT
   tm    CONTROL_INPUTS,#00000010B  ;Test for C     mode select.
   jp    nz,READ_S1                 ;If it is then read S1 inputs
                                    ;Otherwise read S2 (Linear or Stanley)
   ;----- READ DIP SWITCH (S2) SECTION ----------------------------------------------------
                          ;Otherwise read S2 inputs.
READ_S2:
   ;----- READ DIP1-5-----------------------
   and   P2M_COPY,#11110111B        ;Use P2M_COPY to change P23 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P23 to output.
   NOP                              ;The rest of P2M remain unchanged.
   or    P2,#dip_2_1_select         ;Turn P23 on (Hi) to select dip1-5 to be read
   call  READ_P0                    ;Read P0
   and   P2,#11110111B              ;Reset P23 to lo.
   or    P2M_COPY,#00001000B        ;Change P23 back to input pin (Hi Z).
   ld    P2M,P2M_COPY               ;
   ld    DIP_SW_HIBYTE,P0_INPUT     ;Save P0_INPUT to DIP_SW_HIBYTE.
   and   DIP_SW_HIBYTE,#00011111B   ;Mask out non-DIP switch input
   ;----- READ DIP1-5-----------------------

   ;----- READ DIP6-10----------------------
   and   P2M_COPY,#11111011B        ;Use P2M_COPY to change P22 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P22 to output.
   NOP                              ;The rest of P2M remain unchanged.
   or    P2,#dip_2_2_select         ;Turn P22 on (Hi) to select dip6-10 to be read
   call  READ_P0                    ;Read P0
   and   P2,#11111011B              ;Reset P22 to lo.
   or    P2M_COPY,#00000100B        ;Change P22 back to input pin (Hi Z).
   ld    P2M,P2M_COPY               ;
   ld    DIP_SW_LOWBYTE,P0_INPUT    ;Save P0_INPUT to DIP_SW_LOWBYTE.
   and      DIP_SW_LOWBYTE,#00011111B  ;Mask out non-DIP switch input
   ;----- READ DIP6-10----------------------

   ld    ROTATE_COUNTER,#03H        ;Load number of rotations desired (3)
ROTATE_LOWBYTE_S2:
   rlc   DIP_SW_LOWBYTE             ;left rotate lower byte 3 times to bring D6
   dec   ROTATE_COUNTER             ;thru D10 to the left most bits.
   jp    nz,ROTATE_LOWBYTE_S2       ;

   ld    ROTATE_COUNTER,#03H           ;Load number of rotations desired (3)
ROTATE_BOTH_BYTE_S2:
   rlc   DIP_SW_LOWBYTE             ;left rotate lower byte 3 times through carry
   rlc   DIP_SW_HIBYTE              ;left rotate hi byte 3 times through carry.
   dec   ROTATE_COUNTER             ;
   jp    nz,ROTATE_BOTH_BYTE_S2     ;
   jp    EXIT_READ_DIP_SWITCH       ;Finalize and exit subroutine.
   ;----- READ DIP SWITCH (S2) SECTION ----------------------------------------------------

   ;----- READ DIP SWITCH (S1) ------------------------------------------------------------
READ_S1:
   ;----- READ DIP1-5-----------------------
   and   P2M_COPY,#11111101B        ;Use P2M_COPY to change P21 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P21 to output.
   NOP                              ;The rest of P2M remain unchanged.
   or    P2,#dip_1_1_select         ;Turn P21 on (Hi) to select dip1-5 to be read
   call  READ_P0                    ;Read P0
   and   P2,#11111101B              ;Reset P21 to lo.
   or    P2M_COPY,#00000010B        ;Change P21 back to input pin (Hi Z).
   ld    P2M,P2M_COPY               ;
   ld    DIP_SW_HIBYTE,P0_INPUT     ;Save P0_INPUT to DIP_SW_HIBYTE.
   and   DIP_SW_HIBYTE,#00011111B   ;Mask out non-DIP switch input
   ;----- READ DIP1-5-----------------------

   ;----- READ DIP6-10----------------------
   and   P2M_COPY,#11111110B        ;Use P2M_COPY to change P20 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P20 to output.
   NOP                              ;The rest of P2M remain unchanged.
```

```
   or    P2,#dip_1_2_select        ;Turn P20 on (Hi) to select dip6-10 to be read
   call  READ_P0                   ;Read P0
   and   P2,#11111110B             ;Reset P20 to lo.
   or    P2M_COPY,#00000001B       ;Change P20 back to input pin (Hi Z).
   ld    P2M,P2M_COPY              ;
   ld    DIP_SW_LOWBYTE,P0_INPUT   ;Save P0_INPUT to DIP_SW_LOWBYTE.
   and   DIP_SW_LOWBYTE,#00011111B ;Mask out non-DIP switch input
   ;----- READ DIP6-10-----------------------

   ld    ROTATE_COUNTER,#03H       ;Load number of rotations desired (3)
ROTATE_LOWBYTE_S1:
   rlc   DIP_SW_LOWBYTE            ;left rotate lower byte 3 times to bring D6
   dec   ROTATE_COUNTER            ;thru D10 to the left most bits.
   jp    nz,ROTATE_LOWBYTE_S1      ;

   ld    ROTATE_COUNTER,#03H          ;Load number of rotations desired (3)
ROTATE_BOTH_BYTE_S1:
   rlc   DIP_SW_LOWBYTE            ;left rotate lower byte 3 times through carry
   rlc   DIP_SW_HIBYTE             ;left rotate hi byte 3 times through carry.
   dec   ROTATE_COUNTER            ;
   jp    nz,ROTATE_BOTH_BYTE_S1    ;
   ;----- READ DIP SWITCH (S1) ----------------------------------------------------------

   ;----- FINALIZING AND EXIT READ DIP SWITCH ------------------------------------------
EXIT_READ_DIP_SWITCH:
   tm    CONTROL_INPUTS,#00000001B ;Test for " B    mode select.
   jp    nz,ZERO_LOWBYTE           ;If so goto Zeroing out low byte.
CLEAR_LAST_6_BIT:
   and   DIP_SW_LOWBYTE,#11000000B ;Clear out the last 6 bits.
   ret                             ;then exit routine.
ZERO_LOWBYTE:
   clr   DIP_SW_LOWBYTE            ;Clear out lower byte use hi byte only.
   ret                             ;then exit routine.
   ;----- FINALIZING AND EXIT READ DIP SWITCH ------------------------------------------

;==== END READ DIP SWITCH ==========================================================



;==== BEGIN START OF PROGRAM ========================================================
START:
   call  READ_CONTROL_INPUTS       ;Obtain control parameters.
   call  SET_RF_CHANNEL            ;Set P25 accordingly.
   call  READ_DIP_SWITCH           ;Obtain ID code from DIP switch.
   or    IMR,#00000001B            ;Enable IRQ0 (Radio interupt).
   EI                              ;Enable all interupts.
PROGRAM_LOOP:
   call  READ_CONTROL_INPUTS       ;Obtain control parameters.
   call  SET_RF_CHANNEL            ;Set P25 accordingly.
   call  READ_DIP_SWITCH           ;Obtain ID code from DIP switch.
   NOP                             ;
   JP    PROGRAM_LOOP              ;
;==== END START OF PROGRAM ==========================================================
END
```

```
stack_top   equ   07Fh                ;Assigning top of stack at R127.
;*******************************************************************************


;***** INITIALIZE I/O PORTS LABELS ***************************************************
;  Labels/variables related to I/O ports operation.
;*******************************************************************************

debounce_value    equ   0Ah                 ;maximum # for debounce counter.
dip_1_1_select    equ   00000010B           ;Use to turn on P21, shut every thing else off
dip_1_2_select    equ   00000001B           ;Use to turn on P20, shut every thing else off
dip_2_1_select    equ   00001000B           ;Use to turn on P23, shut every thing else off
dip_2_2_select    equ   00000100B           ;Use to turn on P22, shut every thing else off
dip_sw_read       equ   11110000B           ;Use to set P2M to read dip switch


p01m_ini          equ   01000101B           ;P04-P07 = input.
                                            ;Normal external memory (N/A)
                                            ;No P1 for Z86E31.
                                            ;Internal stack only.
                                            ;P03-P00 = input.
p2m_ini           equ   11111111B           ;Default to input (high impedance).
p3m_ini           equ   00000001B           ;P33-P30 = input
                                            ;P37-P34 = output
                                            ;Digital mode for P31, P32.
                                            ;Push-pull active for P2.
p25_lo            equ   11011111B           ;bit5 = 0, Use for freq select
p25_hi            equ   00100000B           ;bit5 = 1, Use for freq select
p2m_in_all        equ   11111111B           ;
pcon_ini          equ   11111110B           ;This is the default value for PCON.
;*******************************************************************************


;***** INTERUPTS  LABELS *************************************************************
;  Labels/variables related to interupt control parameters.
;*******************************************************************************
imr_ini     equ   00000000B           ;Disable all interupt request at start up.
ipr_ini     equ   00010110B           ;Priority IRQ0,2,4,1,5,3.
irq_ini     equ   00000000B           ;Clear out all IRQ initially.
;*******************************************************************************


;***** TIMERs LABELS *****************************************************************
; Labels/variables related to T0 and/or T1 controls. This values are to be used with
; 4 MHz XTAL.
;*******************************************************************************

pre0_100usec      equ   11001001B   ;Value for PRE0 that give T0 resolution of .1msec/count
                                    ;T0 is setup as single pass only. Set bit0 for 'Modulo'.
                                    ;The 6-bit counter = 50D. Whenever this reaches 0 then
                                    ;T0 get decremented by 1. It takes .1 msec for PRE0 to
                                    ;count down from 50 to 0.
t0_1msec          equ   0AH         ;Value for T0 that would time out every 1 msec.
```

```
tmr_ini          equ   00000000B   ;Use to set initial value for TMR's register.
;*******************************************************************************

;***** REGISTERS DEFINITION ****************************************************
;  The section below definning the labels for the particular memory location and/or group.
;*******************************************************************************

;****** ADDITIONAL REGISTERS ***************************************************
;       These registers reside in ERF Bank F of Working Register group 0
;*******************************************************************************
P0       EQU     00H             ;
P1       EQU     01H             ;
P2       EQU     02H             ;
P3       EQU     03H             ;
PCON     EQU     00H             ;Port Configuration register
SMR      EQU     0BH             ;Stop Mode Recovery Register
WDTMR    EQU     0FH             ;Watch Dog Timer Register
;*******************************************************************************

;***** CONTROLLING INPUT REGISTERS *********************************************
;  Using Working Register Group 0 for controlling input parameters such as DIP switch
;  input, mode select, switch mode and frequency select.
;*******************************************************************************
DIP_SW_LOWBYTE    equ   04H         ;Storing input DIP switch from 1-8.
DIP_SW_HIBYTE     equ   05H         ;Storing input DIP switch from 9-10.

DATA_IN_LOWBYTE   equ   06H         ;Storing the decoded lower 8 bit received from RF signal
DATA_IN_HIBYTE    equ   07H         ;Storing the decoded upper 2 bit received from RF signal

CONTROL_INPUTS    equ   08H         ;Store mode select, switch mode selector as defined below
                        ;  bit0  =  0 => Not . B   . mode
                        ;           1 => . B    . mode selected.
                        ;
                        ;  bit1  =  0 => Not . C   . mode.
                        ;           1 => . C    . mode selected.
                        ;
                        ;  bit2  =  0 => Not . A   . mode.
                        ;           1 => . A    . mode selected.
                        ;
                        ;  bit3  =  Not used for ease of coding should be 0.
                        ;
                        ;  bit4  =  Not used for ease of coding should be 0.
                        ;
                        ;  bit5  =  0 => Channel 1 normal switch mode.
                        ;           1 => Channel 1 constant pressure switch mode.
                        ;
                        ;  bit6  =  0 => Channel 2 normal switch mode.
                        ;           1 => Channel 2 constant pressure switch mode.
                        ;
                        ;  bit7  =  Not use for ease of coding should be 0.
USER_FLAGS        equ   09H         ;Store user define flag control bits.
DEBOUNCE_COUNTER  equ   0AH         ;Store debounce value for reading external inputs.
P2M_COPY          equ   0BH         ;Mirror image of P2M register for reading capability.
OLD_INPUT         equ   0CH         ;Use to store previous input of the entire port.
NEW_INPUT         equ   0DH         ;Use to store current input of the entir port.
P0_INPUT          equ   0EH         ;Store P0 input.
ROTATE_COUNTER    equ   0FH         ;Store the # of rotate task to be performed.

;***** WORKING REGISTER GROUP 1 ************************************************
;  Reserve variable for Radio interupts value
;
;***** GROUP 0 *****************************************************************
T0_MULTIPLIER     equ   10H         ;The number of T0 time out without Radio interupts.
                                    ;1 count = 1 msec.
T0_VALUE          equ   11H         ;Store the value of T0 when access.
BIT_COUNTER       equ   12H         ;Store the # of bits received in the data packet.
BIT_0_POSITION_LO equ   13H         ;Store the bit position of the packet data
BIT_0_POSITION_HI equ   14H         ;Store the bit position of the packet data
BIT_1_POSITION_LO equ   15H         ;Store the bit position of the packet data
BIT_1_POSITION_HI equ   16H         ;Store the bit position of the packet data
```

```
DATA_FLAGS          equ   1FH          ;Flags control bit.
                          ;bit0 = 0 => no valid on-time pulse width detected.
                          ;     = 1 => valid on-time pulse width detected.
                          ;
                          ;bit1 = 0 => first edge not detected yet.
                          ;     = 1 => first edge already detected
                          ;
                          ;bit2 = 0 => 10 msec has NOT elapsed from last pulse
                          ;     = 1 => 10 msec has elapsed from last pulse.
                          ;
                          ;bit7 = 0 => Ontime pulse width is for logic 1.
                          ;     = 1 => Ontime pulse width is for logic 0
;===== GROUP 0 ==========================================================================


;===== MACRO DEFINITION ==================================================================
;--------------------WATCH DOG TIMER MACROS ---------------------------------------------
WDT:              MACRO
                  .byte   %5F                     ;Enable Watch dog timer.
;                 .byte   %FF                     ;Disable Watch Dog timer.
                  ENDMAC

;----------------------TIMER CONTROLS MACROS-----------------------------------------
RELOAD_T0:              MACRO
                        or      TMR,#00000001B      ;Set bit0 to reload T0, auto clear
                        ENDMAC                      ;after load by micro.

ENABLE_T0_COUNT:        MACRO
                        or      TMR,#00000010B      ;Set bit1 to enable T0 count.
                        ENDMAC

DISABLE_T0_COUNT:       MACRO
                        and     TMR,#11111101B      ;
                        ENDMAC

RELOAD_T1:              MACRO
                        or      TMR,#00000100B      ;Set bit0 to reload T1, auto clear
                        ENDMAC                      ;after load by micro.

ENABLE_T1_COUNT:        MACRO
                        or      TMR,#00001000B      ;Set bit1 to enable T1 count.
                        ENDMAC

DISABLE_T1_COUNT:       MACRO
                        and     TMR,#11110111B      ;
                        ENDMAC

RESET_MULTIPLIER:       MACRO
                        clr     T0_MULTIPLIER     ;Clear T0 multiplier.
                        ENDMAC
;----------------------TIMER CONTROLS MACROS-----------------------------------------

;----------------------DATA BITS CONTROL MACROS--------------------------------------
RESET_BIT_COUNTER:      MACRO
                        ld      BIT_COUNTER,#01H      ;Set BIT_COUNTER to '1'(1st bit)
                        ENDMAC

RESET_BIT_0_POSITION:   MACRO
                        ld      BIT_0_POSITION_LO,#11111110B;Set BIT_POSITION to the 1st bit.
                        ld    BIT_0_POSITION_HI,#11111111B;
                        ENDMAC

RESET_BIT_1_POSITION:   MACRO
                        ld      BIT_1_POSITION_LO,#00000001B;Set BIT_POSITION to the 1st bit.
                        ld      BIT_1_POSITION_HI,#00000000B;
                        ENDMAC

LOGIC_1:                MACRO
                        or    DATA_FLAGS,#10000000B         ;Set bit7 of flags.
                        ENDMAC
```

```
LOGIC_0:                MACRO
                        and     DATA_FLAGS,#01111111B       ;Clear bit7 of flags.
                        ENDMAC

TEN_MSEC_PASSED:        MACRO
                        or    DATA_FLAGS,#00000100B         ;Set bit2 of flags.
                        ENDMAC

TEN_MSEC_NOT_PASSED:    MACRO
                        and     DATA_FLAGS,#11111011B       ;Clear bit2 of flags.
                        ENDMAC

;----------------------DATA BITS CONTROL MACROS-----------------------------------------
   ENDMAC                           ;after load by micro.
```

What is claimed is:

1. A receiver capable of receiving a plurality of different codes at a plurality of different frequencies, comprising:

an input device for selection among a plurality of different codes and a plurality of different bit patterns;

an antenna for receiving a receiver actuation signal;

digital frequency control circuitry;

a controller for comparing said received receiver actuation signal to said code and bit pattern selections; and

output circuitry for responding to the receipt of a receiver actuation signal that matches said code and bit pattern selections.

2. The receiver of claim 1 wherein said digital frequency control circuitry comprises a signal diode capable of adding and removing discrete components from a bandpass filter.

3. The receiver of claim 2 wherein said input device for selecting among a plurality of different codes is a multi-position switch.

4. The receiver of claim 2 wherein said input device for selecting among a plurality of different bit patterns is a DIP switch.

5. A super-regenerative receiver capable of receiving a plurality of different codes at a plurality of different frequencies, comprising: an input device for selection among a plurality of different codes and a plurality of different bit patterns;

an antenna for receiving a receiver actuation signal;

digital frequency control circuitry;

a controller for comparing said received receiver actuation signal to said code and bit pattern selections; and

output circuitry for responding to the receipt of a receiver actuation signal that matches said code and bit pattern selections.

6. A radio frequency receiver for receiving a plurality of actuation signals from a movable barrier operator transmitter, each receiver being capable of receiving a plurality of coded signals at a plurality of different frequencies, comprising:

first and second user-selectable input devices for selecting a specified code and a specified frequency for receiving said actuation signals;

a controller coupled to said input devices for processing said code and frequency selections and outputting data responsive to said input; and

receiver circuitry responsive to said controller output data for receiving particular actuation signals at one frequency and receiving particular other actuation signals at another frequency.

7. The radio frequency receiver of claim 6, wherein said first user-selectable input device comprises a multi-position switch which determines a particular code to be received as said actuation signal based upon the position of said multi-positioned switch.

8. The radio frequency receiver of claim 7, wherein said second user-selectable input device comprises a dual in-line packaged switch having a plurality of inner switches which determine a particular bit sequence to be received as said actuation signal based upon the position of said plurality of inner switches.

9. The radio frequency receiver of claim 8, wherein said controller processes the code and bit sequence selections from said input devices and outputs data according to said input to said receiver circuitry causing said receiver circuitry to receive particular data at one frequency and other data at another frequency.

10. A method of digitally controlling the frequency of a receiver comprising the steps of:

providing a bandpass filter;

providing a signal diode connecting additional discrete components to said bandpass filter;

providing a controller for controlling the operation of said signal diode to alter the discrete component makeup of the bandpass filter to adjust frequency; and

outputting signals to said diode to alter the bandpass filter frequency.

11. A method of receiving a receiver actuating signal comprising the steps of:

providing a receiver having multiple input devices coupled to a microprocessor and receiver circuitry;

adjusting said receiver circuitry to receive a particular code at a particular frequency based on the position of said multiple input devices and output from said microprocessor; and

receiving said receiver actuating signal.

12. The method of claim 11 wherein one of said multiple input devices is a multi-positioned switch which determines the code to be received as said receiver actuating signal based upon the position of said multi-positioned switch.

13. The method of claim 12 wherein another of said multiple input devices is a dual in-line packaged switch having multiple inner switches which determines a bit pattern to be received as said receiver actuating signal based upon the position of said inner switches.

ABSTRACT OF THE DISCLOSURE

A receiver capable of receiving a plurality of different codes at a plurality of different frequencies is set forth herein. The receiver offers digital frequency control by using a controller driven signal diode to add or remove capacitance to or from a band pass filter circuit, thereby altering the frequency the bandpass filter is setup to receive. Input devices allow the receiver code, bit pattern, and relay output to be selected among a plurality of different codes, bit patterns, and relay output types. The receiver processes the selections, determines what frequency receiver actuation signal is to be received, and alters the bias of the signal diode to adjust the bandpass filter frequency accordingly.

Fig. 1

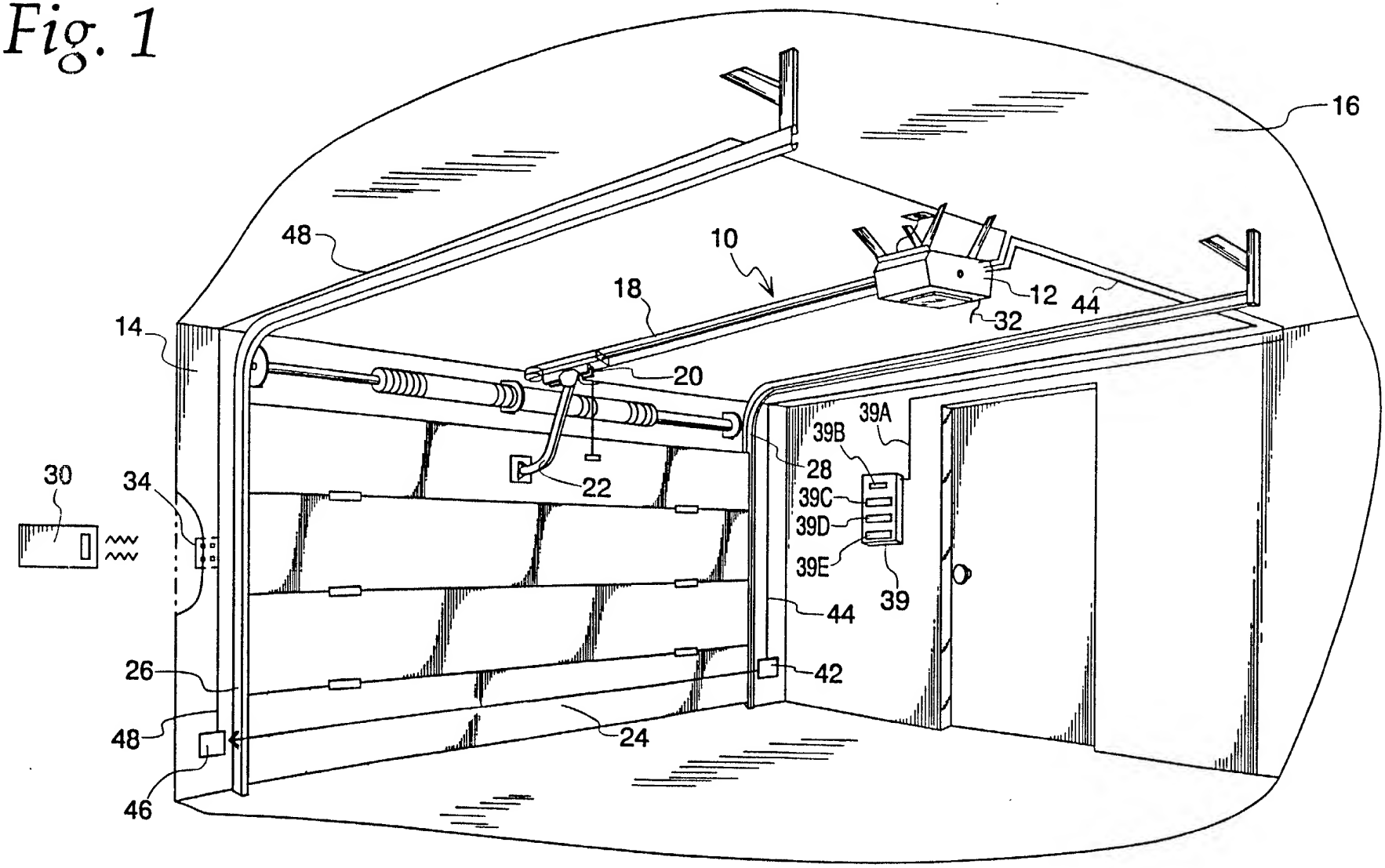
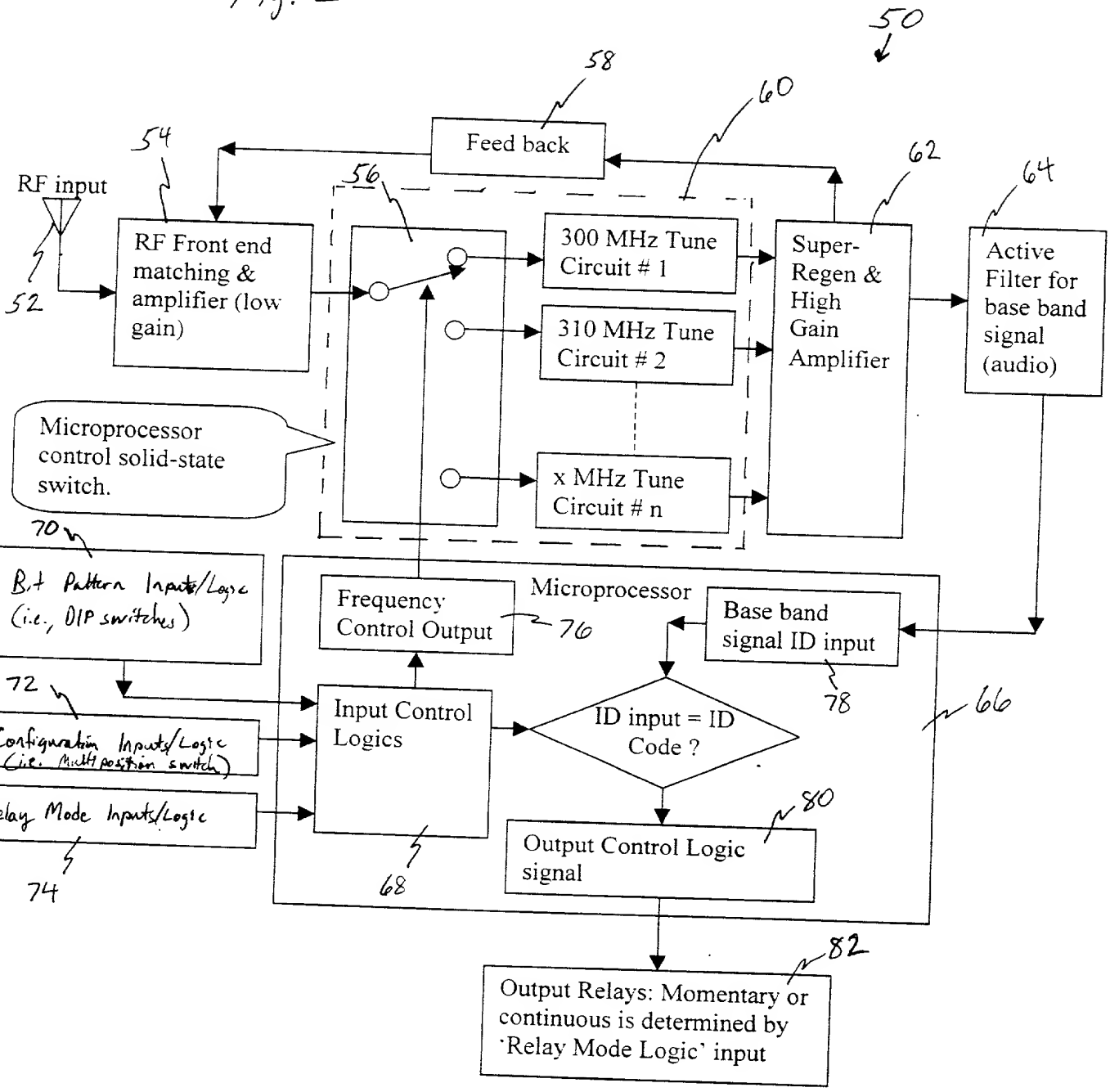


Fig. 2


50
58
60
Feed back
54
56
62
64
RF input
52
RF Front end matching & amplifier (low gain)
300 MHz Tune Circuit # 1
310 MHz Tune Circuit # 2
x MHz Tune Circuit # n
Super-Regen & High Gain Amplifier
Active Filter for base band signal (audio)
Microprocessor control solid-state switch.
70
R,t Pattern Inputs/Logic (i.e., DIP switches)
72
Configuration Inputs/Logic (i.e. Multi position switch)
Relay Mode Inputs/Logic
74
Frequency Control Output
76
Microprocessor
Base band signal ID input
78
66
Input Control Logics
68
ID input = ID Code ?
80
Output Control Logic signal
82
Output Relays: Momentary or continuous is determined by 'Relay Mode Logic' input

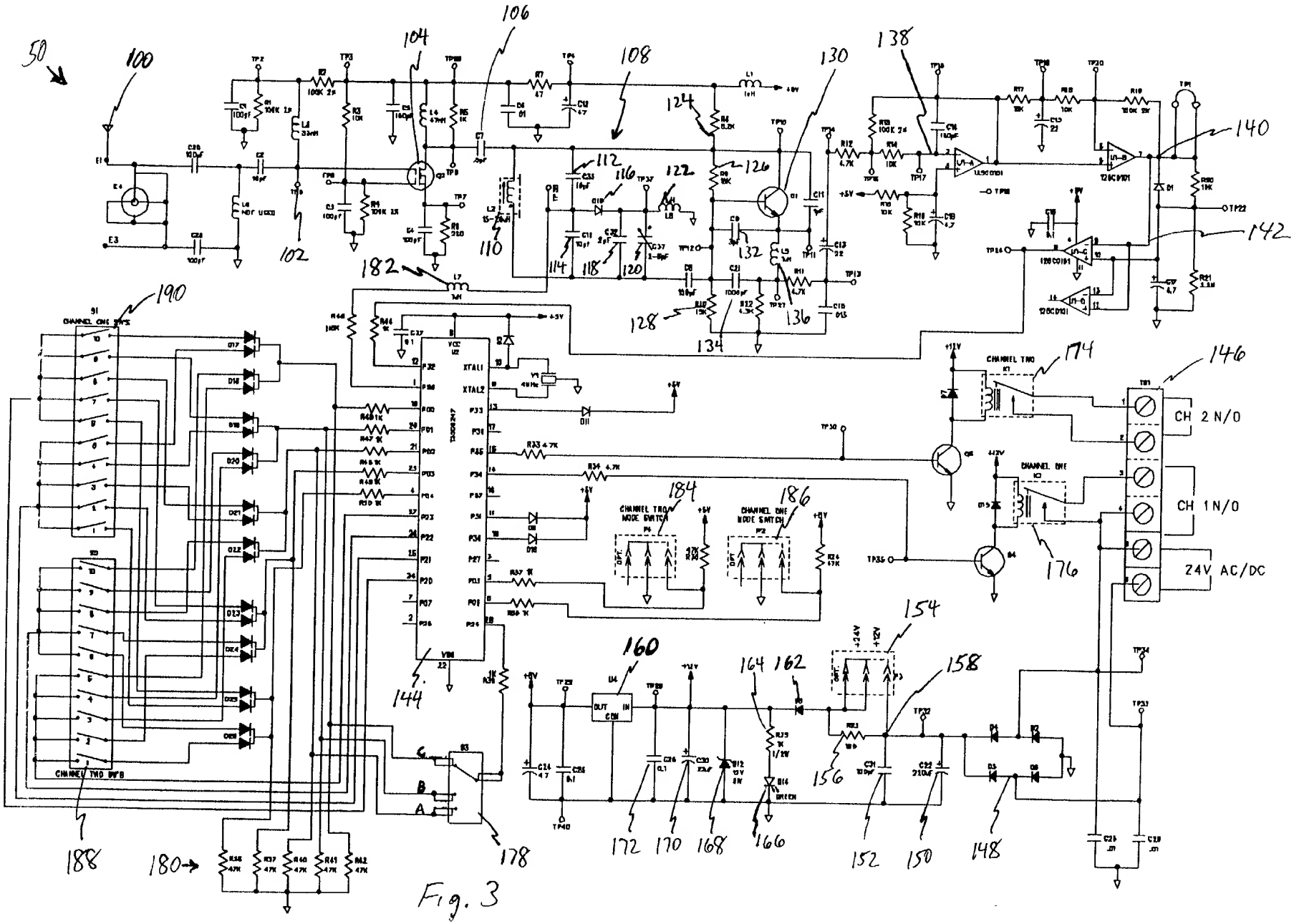


Fig. 3

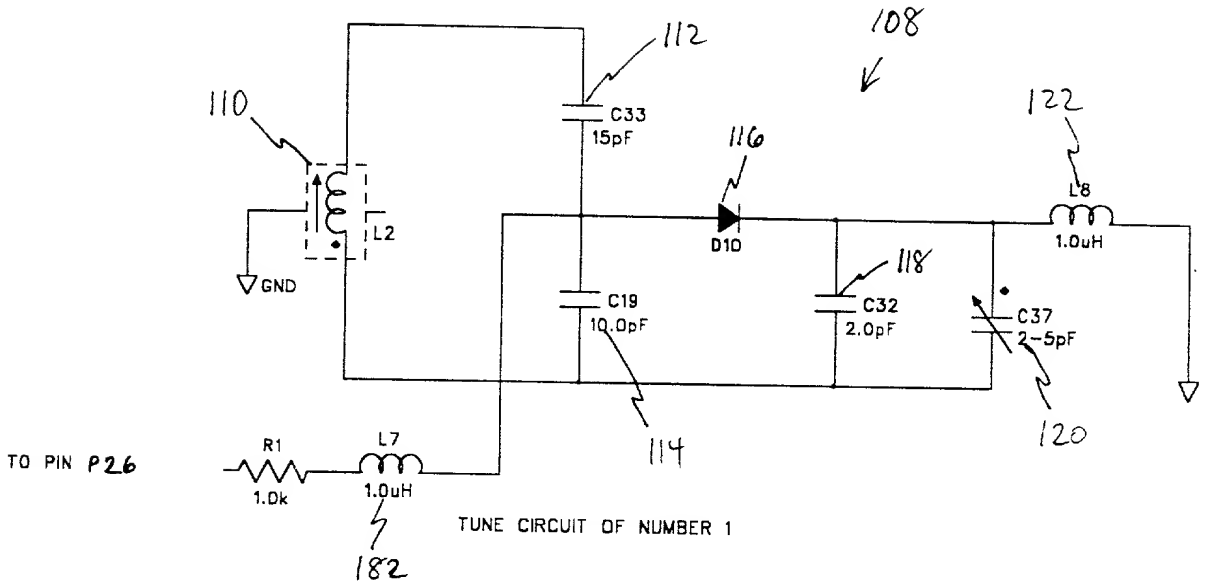

108
112
110
C33
15pF
116
122
L8
1.0uH
L2
D10
GND
118
C19
10.0pF
C32
2.0pF
C37
2-5pF
114
120
TO PIN P26
R1
1.0k
L7
1.0uH
182
TUNE CIRCUIT OF NUMBER 1

Fig. 4A

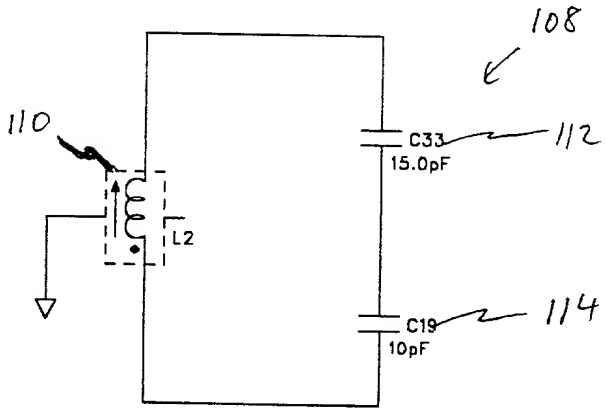

108
110
C33
15.0pF
112
L2
C19
10pF
114
TUNED CIRCUIT NUMBER 2

Fig. 4B

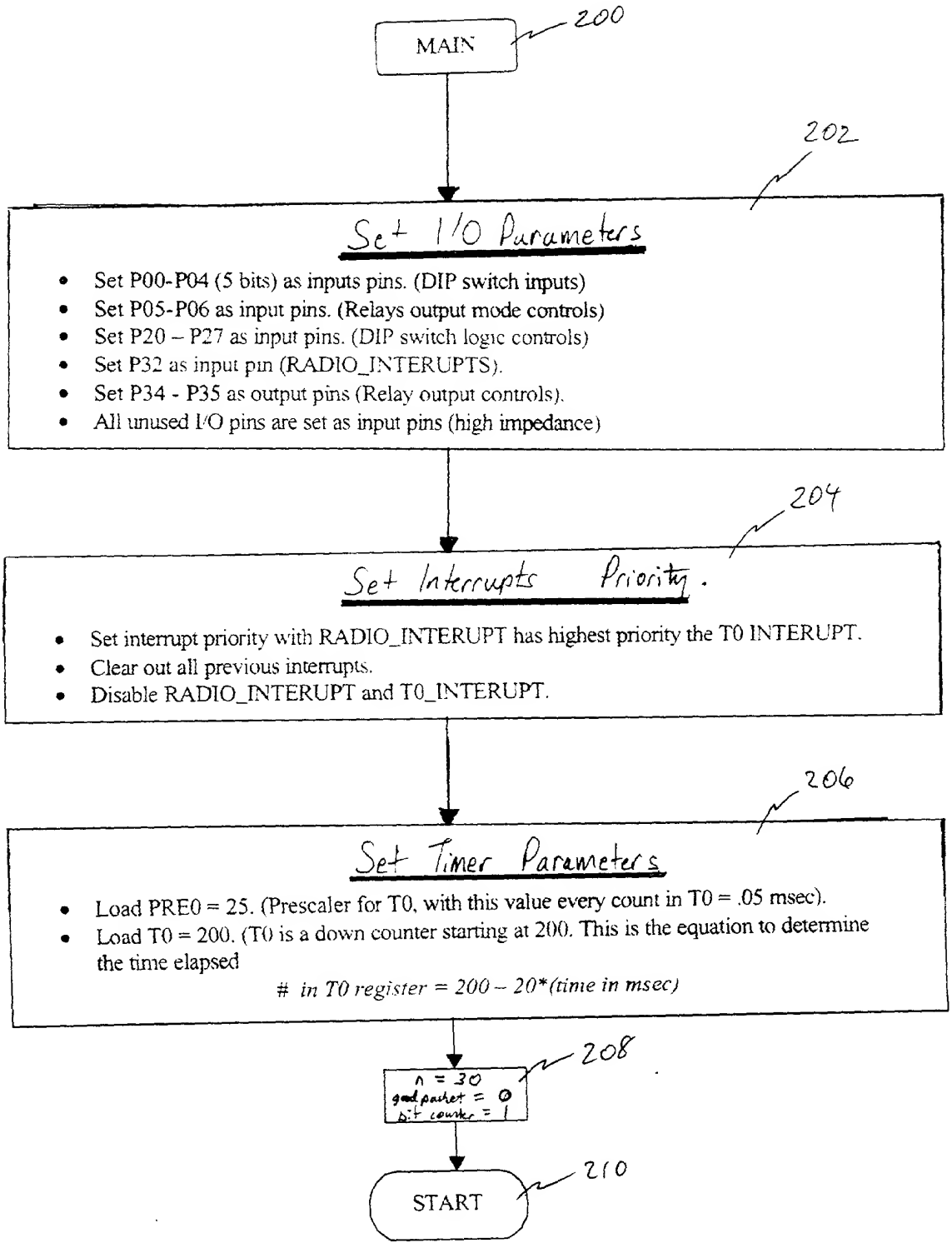

MAIN
200
202
Set I/O Parameters
Set P00-P04 (5 bits) as inputs pins. (DIP switch inputs)
Set P05-P06 as input pins. (Relays output mode controls)
Set P20 – P27 as input pins. (DIP switch logic controls)
Set P32 as input pin (RADIO_INTERUPTS).
Set P34 - P35 as output pins (Relay output controls).
All unused I/O pins are set as input pins (high impedance)
204
Set Interrupts Priority.
Set interrupt priority with RADIO_INTERUPT has highest priority the T0 INTERUPT.
Clear out all previous interrupts.
Disable RADIO_INTERUPT and T0_INTERUPT.
206
Set Timer Parameters
Load PRE0 = 25. (Prescaler for T0, with this value every count in T0 = .05 msec).
Load T0 = 200. (T0 is a down counter starting at 200. This is the equation to determine the time elapsed
in T0 register = 200 – 20*(time in msec)
208
n = 30
good packet = 0
bit counter = 1
START
210

Fig. 5A

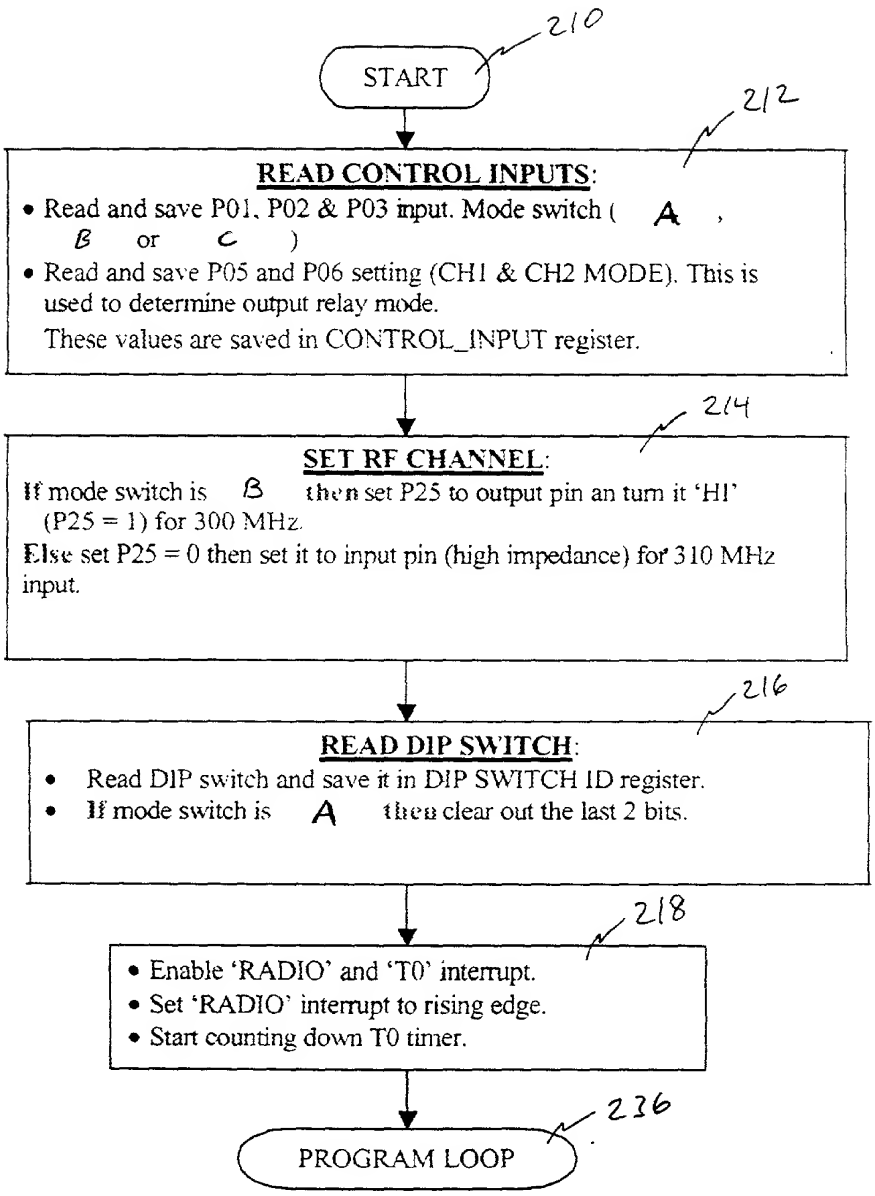

210
START
212
READ CONTROL INPUTS:
• Read and save P01, P02 & P03 input. Mode switch (A , B or C)
• Read and save P05 and P06 setting (CH1 & CH2 MODE). This is used to determine output relay mode.
These values are saved in CONTROL_INPUT register.
214
SET RF CHANNEL:
If mode switch is B then set P25 to output pin an turn it 'HI' (P25 = 1) for 300 MHz.
Else set P25 = 0 then set it to input pin (high impedance) for 310 MHz input.
216
READ DIP SWITCH:
• Read DIP switch and save it in DIP SWITCH ID register.
• If mode switch is A then clear out the last 2 bits.
218
• Enable 'RADIO' and 'T0' interrupt.
• Set 'RADIO' interrupt to rising edge.
• Start counting down T0 timer.
236
PROGRAM LOOP

Fig. 5B

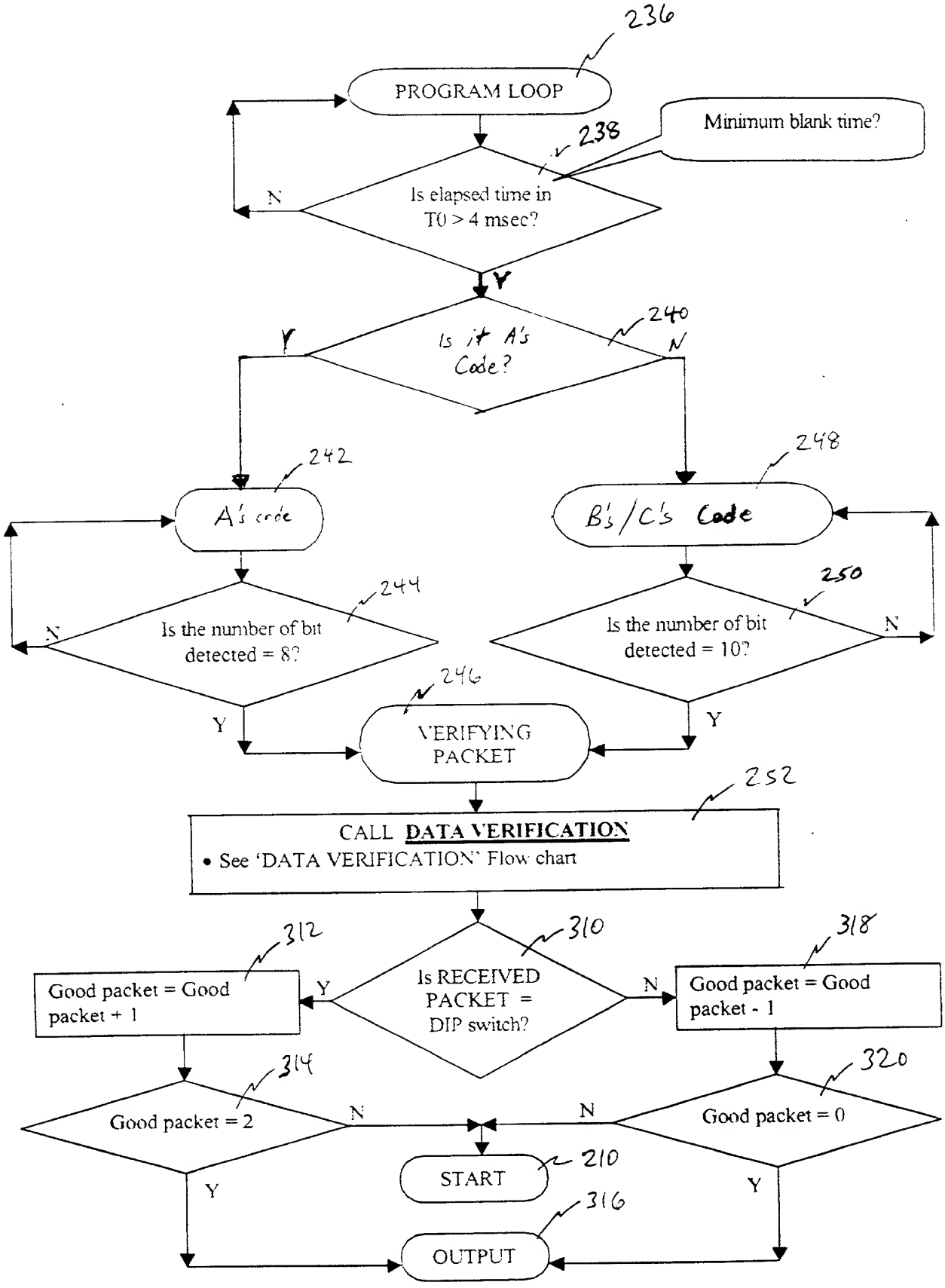

236
PROGRAM LOOP
238
Minimum blank time?
Is elapsed time in T0 > 4 msec?
N
Y
240
Is it A's Code?
Y
N
242
A's code
248
B's/C's Code
244
Is the number of bit detected = 8?
N
250
Is the number of bit detected = 10?
N
Y
Y
246
VERIFYING PACKET
252
CALL DATA VERIFICATION
• See 'DATA VERIFICATION' Flow chart
310
Is RECEIVED PACKET = DIP switch?
312
Good packet = Good packet + 1
Y
N
318
Good packet = Good packet - 1
314
Good packet = 2
N
N
320
Good packet = 0
Y
Y
210
START
316
OUTPUT

Fig. 5C

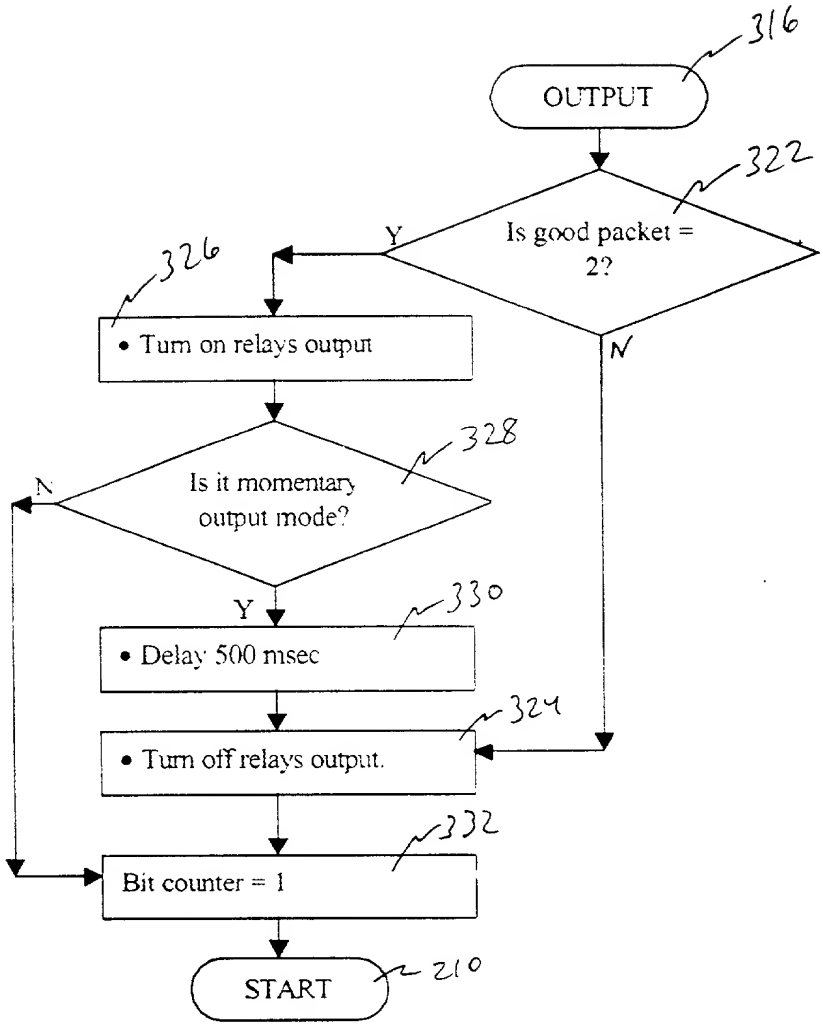

316
OUTPUT
322
Is good packet = 2?
Y
N
326
• Turn on relays output
328
Is it momentary output mode?
N
Y
330
• Delay 500 msec
324
• Turn off relays output.
332
Bit counter = 1
START
210

Fig. 5D

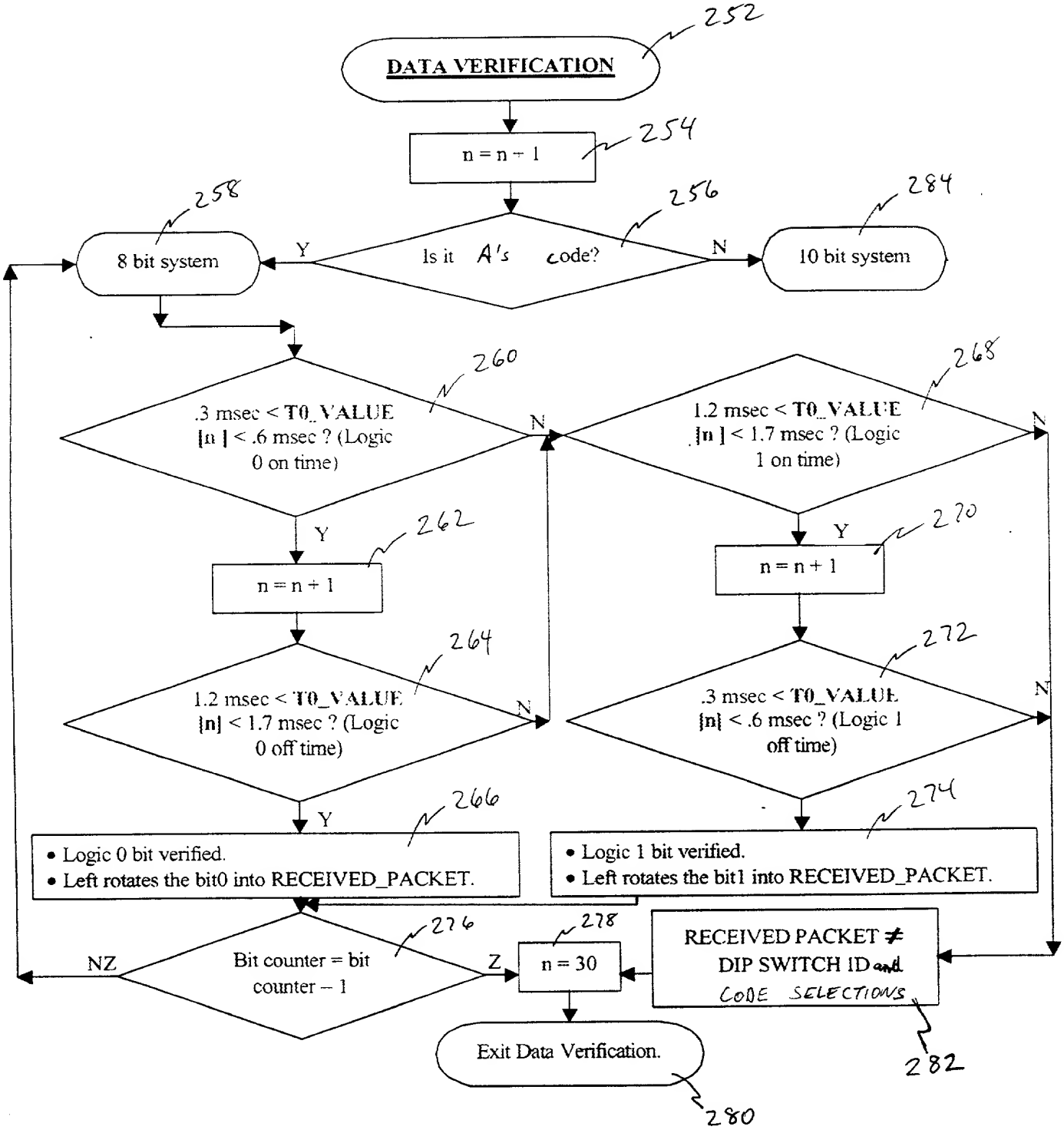

252
DATA VERIFICATION
254
n = n + 1
256
Is it A's code?
Y
N
258
8 bit system
284
10 bit system
260
.3 msec < T0_VALUE |n| < .6 msec ? (Logic 0 on time)
N
Y
262
n = n + 1
264
1.2 msec < T0_VALUE |n| < 1.7 msec ? (Logic 0 off time)
N
Y
266
• Logic 0 bit verified.
• Left rotates the bit0 into RECEIVED_PACKET.
268
1.2 msec < T0_VALUE |n| < 1.7 msec ? (Logic 1 on time)
N
Y
270
n = n + 1
272
.3 msec < T0_VALUE |n| < .6 msec ? (Logic 1 off time)
N
274
• Logic 1 bit verified.
• Left rotates the bit1 into RECEIVED_PACKET.
276
Bit counter = bit counter – 1
NZ
Z
278
n = 30
RECEIVED PACKET ≠ DIP SWITCH ID and CODE SELECTIONS
282
Exit Data Verification.
280

Fig. 5E

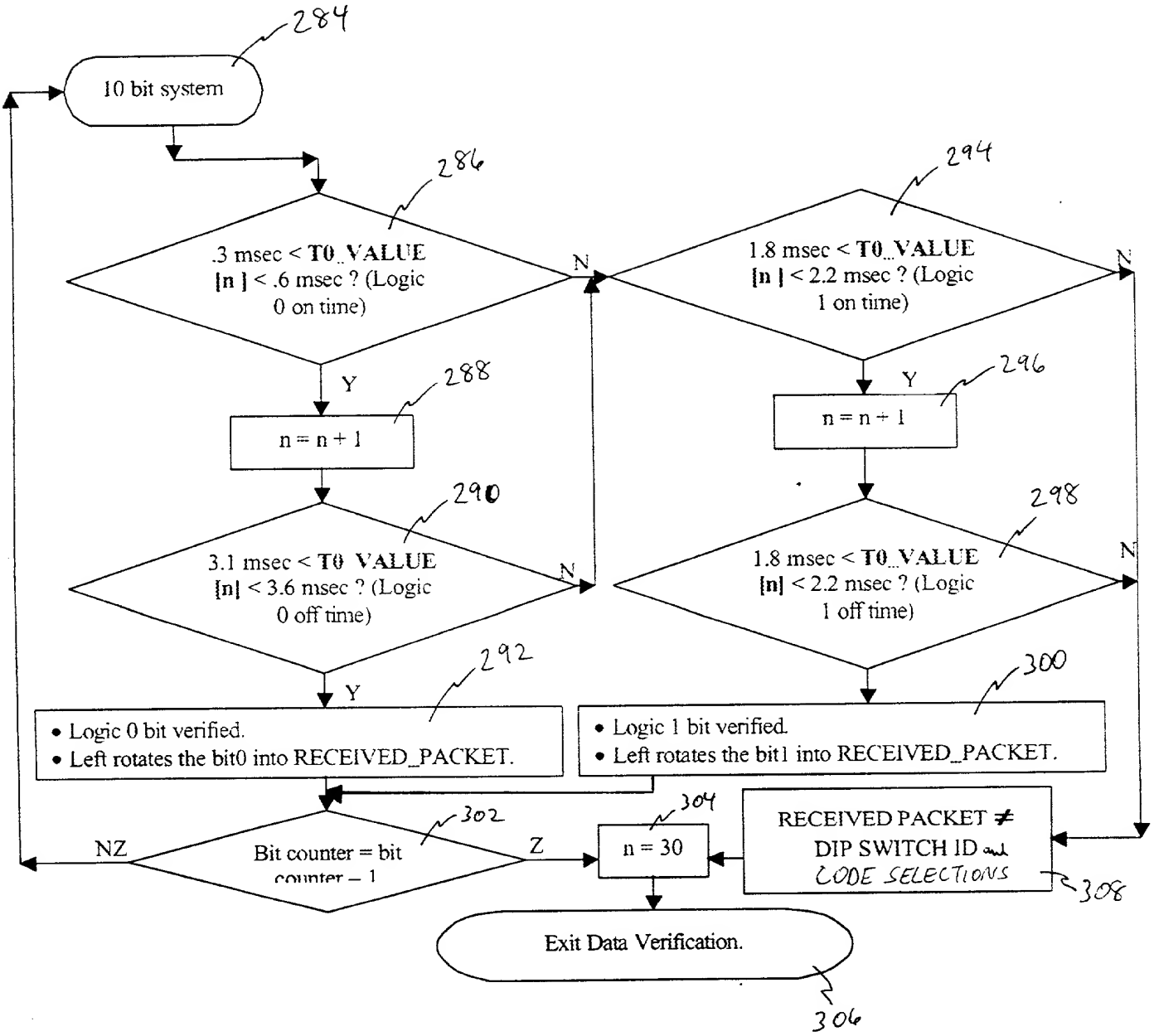

284
10 bit system
286
.3 msec < T0_VALUE [n] < .6 msec ? (Logic 0 on time)
N
Y
288
n = n + 1
290
3.1 msec < T0_VALUE [n] < 3.6 msec ? (Logic 0 off time)
N
Y
292
• Logic 0 bit verified.
• Left rotates the bit0 into RECEIVED_PACKET.
294
1.8 msec < T0_VALUE [n] < 2.2 msec ? (Logic 1 on time)
N
Y
296
n = n + 1
298
1.8 msec < T0_VALUE [n] < 2.2 msec ? (Logic 1 off time)
N
300
• Logic 1 bit verified.
• Left rotates the bit1 into RECEIVED_PACKET.
302
Bit counter = bit counter – 1
NZ
Z
304
n = 30
RECEIVED PACKET ≠ DIP SWITCH ID and CODE SELECTIONS
308
Exit Data Verification.
306

Fig. 5F

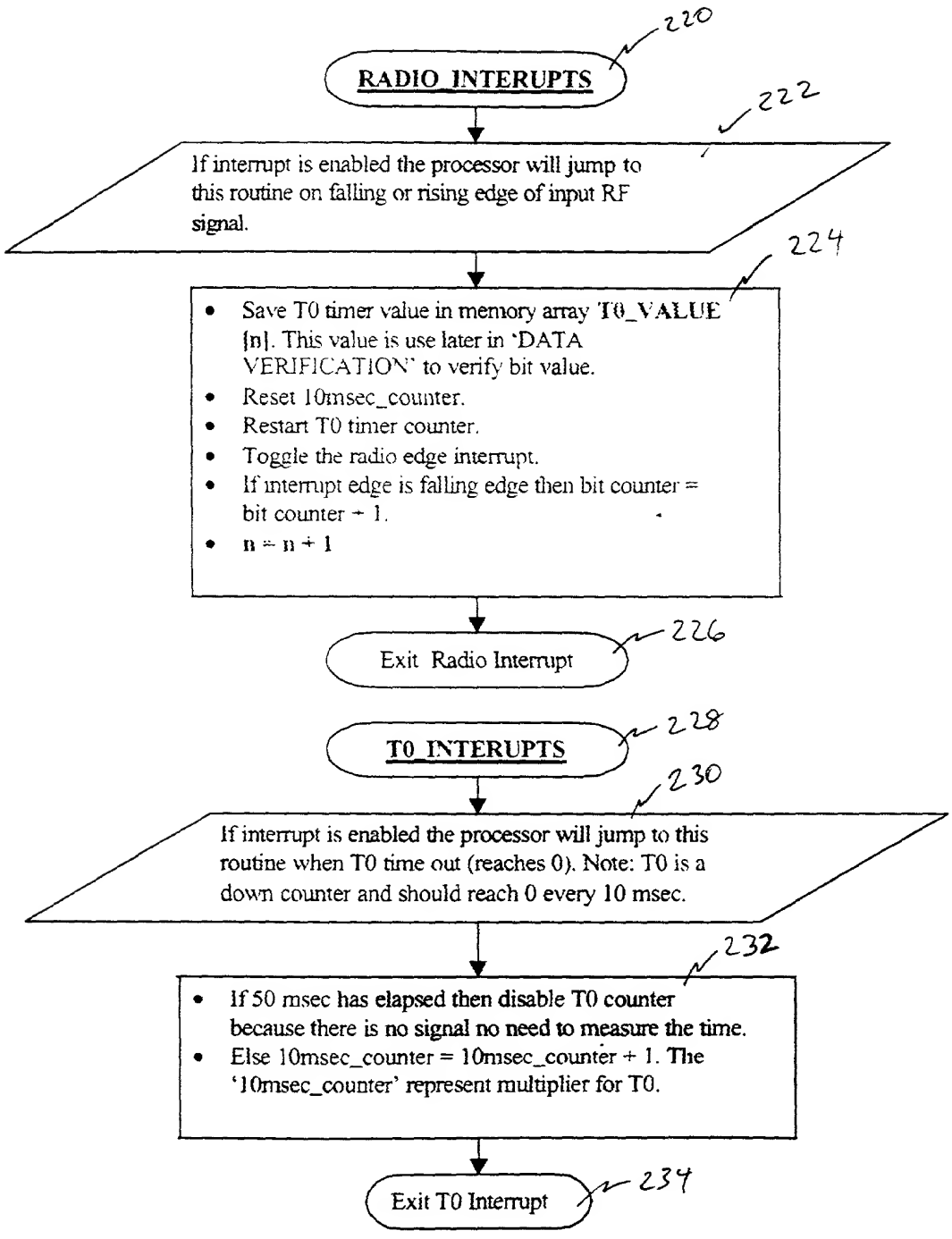

220
RADIO_INTERUPTS
222
If interrupt is enabled the processor will jump to this routine on falling or rising edge of input RF signal.
224
• Save T0 timer value in memory array T0_VALUE [n]. This value is use later in 'DATA VERIFICATION' to verify bit value.
• Reset 10msec_counter.
• Restart T0 timer counter.
• Toggle the radio edge interrupt.
• If interrupt edge is falling edge then bit counter = bit counter + 1.
• n = n + 1
226
Exit Radio Interrupt
228
T0_INTERUPTS
230
If interrupt is enabled the processor will jump to this routine when T0 time out (reaches 0). Note: T0 is a down counter and should reach 0 every 10 msec.
232
• If 50 msec has elapsed then disable T0 counter because there is no signal no need to measure the time.
• Else 10msec_counter = 10msec_counter + 1. The '10msec_counter' represent multiplier for T0.
234
Exit T0 Interrupt

Fig. 5G.

DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION

☒ Declaration Submitted With Initial Filing ☐ Declaration Submitted After Initial Filing

Attorney Docket No.: 68702

First Named Inventor: Nguyen et al.

Application Number:

Filing Date: April 25, 2000

Group Art Unit:

Examiner Name:

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR RECEIVING A PLURALITY OF DIFFERENT CODES AT A PLURALITY OF DIFFERENT FREQUENCIES

(Title of Invention)

the specification of which:

(X) is attached hereto, or

( ) was filed by an authorized person on my behalf on _____ (Date)
as United States Application Number _____
or PCT International Application Number _____,
and was amended on _____ (if applicable).
(Date)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119(a)-(d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT international application which designated at least one country other than the United States of America, listed below, and I have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or any PCT international application, on this invention filed by me or my legal representatives or assigns and having a filing date before that of the application on which priority is claimed:

Prior Foreign Application Number(s)	Country	Foreign Filing Date	Priority Not Claimed	Certified Copy Attached Yes	No
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐

☐ Additional foreign application numbers are listed on a supplemental priority data sheet attached hereto.

I hereby claim the benefit under Title 35, United States Code, §119(e) of any United States provisional application(s) listed below:

Provisional Application Number(s)	Provisional Application Filing Date

☐ Additional provisional application numbers are listed on a supplemental priority data sheet attached hereto.

I hereby claim the benefit under Title 35, United States Code, §120, of any prior United States application(s), or under §365(c) of any PCT international application(s) designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application(s) in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose all information known by me to be material to patentability as defined in Title 37, Code of Federal Regulations, §1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Prior U.S. Application Number	Prior PCT International Application Number	Filing Date of U.S. or PCT International Application	Patent Number (if applicable)

☐ Additional U.S. or PCT international application numbers are listed on a supplemental priority data sheet attached hereto.

As a named inventor, I hereby appoint the practitioners associated with Customer Number 22242, with full power of substitution and revocation, to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith, and request that all correspondence and telephone calls in respect to this application be directed to FITCH, EVEN, TABIN & FLANNERY, Suite 1600, 120 South LaSalle

Street, Chicago, Illinois 60603-3406, Telephone No. (312) 577-7000, Facsimile No. (312) 577-7007, CUSTOMER NUMBER 22242.

I hereby declare that all statements made herein of my own knowledge are true, and that all statements made herein on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity or enforceability of the application or any patent issued thereon.

Full name of sole or one joint inventor: Hung Nguyen
(Given names first, with Family name last)

Inventor's signature: _____

Date: _____

Residence: Patchogue, New York
(City and State for U.S. Residents; City and Country for others)

Post Office Address: 52 Academy Street

_____

_____

Citizenship: U.S.A.

Full name of sole or one joint inventor: Bernard Wojciak
(Given names first, with Family name last)

Inventor's signature: _____

Date: _____

Residence: Naperville, Illinois
(City and State for U.S. Residents; City and Country for others)

Post Office Address: 1925 Vassar

_____

_____

Citizenship: U.S.A.

Attorney Docket No. 68702

Supplemental Data Priority Sheet

☐ Additional foreign application numbers:

Prior Foreign Application Number(s)	Country	Foreign Filing Date	Priority Not Claimed	Certified Copy Attached Yes	Certified Copy Attached No
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐

☐ Additional provisional application numbers:

Provisional Application Number(s)	Provisional Application Filing Date

☐ Additional U.S. or PCT international application numbers:

Prior U.S. Application Number	Prior PCT International Application Number	Filing Date of U.S. or PCT International Application	Patent Number (if applicable)